2024 American Control Conference (ACC)
July 8-12, 2024. Toronto, Canada

Adaptive Low-Rank Tensor Approximation based on Mixed-Integer Representations

Zhi Xu, Chaoying Pei, and Ran Dai

Abstract— **Tensor structures are fundamental in addressing the intricate challenges posed by high-dimensional data across a spectrum of scientific and computational domains. Within this context, low-rank tensor approximation plays a pivotal role in enhancing data processing efficiency. This paper develops a novel adaptive low-rank tensor approximation method by introducing mixed-integer representations to identify an appropriate low-rank approximation for high-dimensional tensors. The approach takes into consideration both tensor rank determination and approximation accuracy, leveraging binary variables to represent tensor ranks that will be optimized as unknowns with the tensor arrays. By integrating the alternating least squares technique with the truncation method, the integrated algorithm effectively achieves a proper low-rank tensor with high approximation accuracy. To substantiate its efficacy and efficiency in solving tensor approximation problems, the paper provides extensive simulation results and analysis.**

I. Introduction

In recent years, the exponential growth in data size across multiple fields has prompted the demand for more effective and efficient data processing structures. Tensors, with their inherent advantages of versatility and efficiency, have attracted growing interest across various scientific and computational domains. Consequently, their applications have rapidly expanded to encompass diverse fields, such as data analysis [1], machine learning [2], and image processing [3], among others.

Tensors appear as an extension of vectors and matrices — a multidimensional array indexed by three or more indices, whereas matrices are indexed by two. Consequently, tensor algebra often exhibits similarities to matrix algebra. Nevertheless, notable distinctions exist between these two algebras. For instance, while low-rank tensor factorization tends to be essentially unique under mild conditions, determining the tensor rank is NP-hard. On the other hand, in contrast to matrices, tensors often involve significantly larger datasets, leading to a heavier computational load [4]. One of the most prominent domains where tensors find applications is in numerical computation [5], [6]. For example, a multivariate function $f(x_1, x_2, \ldots, x_d)$ with dimension d can be approximated by a mode-d tensor through grid-based sampling. In this case, each entry of the tensor contains the value of function f at the corresponding position. The function f itself may represent a high-dimensional function or a solution to high-dimensional partial differential equations (PDE). As the problem dimension increases, problem size and complexity grow exponentially with d, causing numerical computation or even storage of data intractable. To address these challenges, one of the most effective approaches is to seek low-rank approximations of large-scale tensors. However, recall that determining the rank of large-scale tensors is a computationally NP-hard problem, approximating a high-dimensional tensor without precise knowledge of its rank remains another ongoing challenge.

In this context, tensor decomposition techniques play a pivotal role, offering various tensor formats to address high-dimensional challenges. For example, Canonical Polyadic (CP) format enables the efficient decomposition of complex and high-dimensional data and operations into a combination of one-dimensional components [7], [8]. This process is also known as *separated representation* [9], [10]. As an outcome, the complexity scales linearly rather than exponentially increase with dimensionality. Due to this advantage, CP decomposition is widely applied in model order reduction for large-scale nonlinear problems [11]–[14]. Different from CP format, Tucker format decomposes a tensor array into one small core tensor and a set of matrices to reveal the algebraic structure of a given tensor [15]. Nevertheless, the need to manually predefine Tucker ranks along all modes can pose practical difficulties [16]. Tensor Train (TT) format [17], [18], as applied by Richter et al. [19] has found utility in approximating solutions to high-dimensional parabolic PDEs. It has demonstrated superior performance compared to state-of-the-art neural network-based approaches, trading a higher compression rate with the simplicity of one-dimensional separated representation of CP decomposition. Tree tensor network (TTN) factorizes a high-rank tensor into an acyclic network (tree) formed of rank-3 tensors [20]. This format has gained success in applications of quantum many-body physics [21]. However, selecting appropriate decomposition for tensor trees often relies on heuristics and problem-specific context [22].

In this paper, we focus on tensor decomposition in CP format due to the simplicity and interpretability of separated representation. Obtaining a tensor approximation through CP tensor decomposition necessitates specifying the rank for a tensor to be decomposed. However, if the rank is chosen too large, the resulting approximated tensor may retain excessive scale. Conversely, estimating a rank significantly lower than the actual value can negatively impact the accuracy of the tensor approximation. Therefore, a balance between computational cost and precise approximation is necessary for tensor decomposition. Following this idea, this work focuses on developing a computationally feasible low-rank tensor approximation algorithm, aiming to adaptively determine a precise estimate of tensor rank and subsequently construct the approximation.

The alternating least squares (ALS) method is a widely adopted approach for computing CP decomposition in existing literature [23]. For instance, Beylkin et al. [9] approximated the solution of high-dimensional Schrodinger equation in the CP decomposition form via ALS. Similarly, Sun and Kumar [24] combined ALS based CP decomposition approach with Chebyshev spectral differentiation to solve high-dimensional stationary Fokker-Planck equations. Boelens et al. [25] proposed a parallel algorithm based on canonical and hierarchical numerical tensor methods that combine ALS and hierarchical singular value decomposition to solve the Boltzmann and Fokker–Planck equations. In the field of optimal control, Horowitz et al. [26] proposed an ALS-based tensor representation method to address the curse of dimensionality in solving the high-dimensional linear Hamilton-Jacobi-Bellman (HJB) equation. On this basis, Stefansson and Leong [27] proposed an improved sequential ALS method to resolve the ill-conditioning issue of the original ALS method by introducing boundary condition re-scaling

Zhi Xu, chaoying Pei, and Ran Dai are with the School of Aeronautic and Astronautics, Purdue University, West Lafayette, IN 47907, USA. Emails: xu1710@purdue.edu, peic@purdue.edu and randai@purdue.edu

and then computing the solution sequentially.

Furthermore, the ALS method is also a widely employed technique for achieving low-rank approximations. A commonly practiced strategy involves starting with a small rank and progressively increasing it by introducing a randomly initialized one-dimensional component [9], [24], [26], [27]. In each iteration, the tensor approximation with a new rank needs to be recomputed through ALS, until the approximation error is smaller than a threshold. Although ALS has been proven to be a relatively efficient algorithm for tensor decomposition [23], a large number of iterations may still lead to a heavy computational load, especially for large-scale problems with an inherent high rank. Besides, taking the approximation error as a stop criterion is not straightforward for balancing between the tensor rank and approximation accuracy. Alternative low-rank tensor approximation methods include singular value decomposition [28], projection method [29], and conjugate gradient method [30], just to name a few. However, none of these approaches have an explicit representation of the rank of the decomposed tensor.

The contribution of the proposed method in this paper can be summarized in three key aspects.

- Through the incorporation of binary variables, the output tensor's rank can be rigorously ensured as an integer value, thereby mitigating potential errors in the tensor reconstruction process. The introduction of binary variables for tensor rank representation also facilitates the low-rank approximation computation, which formulates the low-rank tensor approximation problem as a mixed-integer optimization problem.
- By simultaneously minimizing the tensor's rank and the approximation error, the output tensor accurately estimates its minimally required rank while maintaining high accuracy in representing the original data.
- Combining the ALS method with truncation techniques serves to reduce computational complexity while enhancing convergence speed.

The paper is structured as follows: In Section II, we introduce the notation and provide the necessary tensor preliminaries. Section III presents the problem formulation, while Section IV details our proposed adaptive low-rank tensor approximation method. In Section V, we present and analyze the results obtained by applying our method to a range of low-rank tensor approximation problems. Finally, Section VI offers our conclusions.

II. Notation and Preliminaries

In this section, we aim to provide a concise introduction to tensors, encompassing their fundamental properties, operations, and essential concepts such as tensor rank. A tensor can be concisely represented as $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times \dots n_d}$, where n_1, n_2, $\cdots$, n_d denote the tensor's dimensions, with d being the terminology used to describe the tensor's order or mode. For example, a tensor with a mode of 1 corresponds to a vector, while a tensor with a mode of 2 corresponds to a matrix.

To enhance the reader's understanding, we will now introduce various tensor operations.

Tensor addition: Tensor Addition: When faced with two tensors, denoted as $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$, sharing identical dimensions, the process of adding them element-wise results in the creation of a new tensor denoted as $\boldsymbol{\mathcal{C}}$, written as

$$\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{B}}. \tag{1}$$

Scalar multiplication: A tensor $\mathbf{A}$ can be multiplied by a scalar k, achieved by multiplying each element of the tensor by the scalar value k, expressed as

$$\boldsymbol{\mathcal{B}} = k \cdot \boldsymbol{\mathcal{A}}. \tag{2}$$

Inner product: The inner product of two tensors of the same size, denoted as $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \in \mathbb{R}^{n_1 \times n_2 \times \dots n_d}$, is obtained by summing the products of their corresponding elements, expressed as

$$\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} a_{i_1 i_2 \cdots i_d} b_{i_1 i_2 \cdots i_d}, \tag{3}$$

where $a_{i_1 i_2 \cdots i_d}$ and $b_{i_1 i_2 \cdots i_d}$ are elements in tensors $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$, respectively. Then naturally, it follows that $\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{A}} \rangle = \|\boldsymbol{\mathcal{A}}\|$.

Outer product: In contrast to the inner product, the outer product involving two tensors, $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times \dots n_d}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{m_1 \times m_2 \times \dots m_p}$, results in the creation of a new tensor, denoted as $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{n_1 \times n_2 \times \dots n_d \times m_1 \times m_2 \times \dots m_p}$. The elements of $\boldsymbol{\mathcal{C}}$ are computed by multiplying each element of $\boldsymbol{\mathcal{A}}$ with each element of $\boldsymbol{\mathcal{B}}$, resulting in a tensor with expanded dimensions, expressed as:

$$\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \odot \boldsymbol{\mathcal{B}} \tag{4}$$

where $\boldsymbol{\mathcal{C}}_{i_1 i_2 \cdots i_d j_1 j_2 \cdots j_p} = \boldsymbol{\mathcal{A}}_{i_1 i_2 \cdots i_d} \cdot \boldsymbol{\mathcal{B}}_{j_1 j_2 \cdots j_p}$.

Tensor transpose: Tensor transpose transforms the tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times \dots n_d}$ into a new one $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{n_d \times n_{d-1} \times \cdots \times n_1}$, with dimensions rearranged in reverse order. The mathematical representation is

$$\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{A}}^T. \tag{5}$$

Tensor Inverse: Tensor inversion, denoted as $\boldsymbol{\mathcal{A}}^{-1}$, is the operation of finding the inverse of a square tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n \times n}$. The inverse tensor, $\boldsymbol{\mathcal{A}}^{-1}$, satisfies the equation

$$\boldsymbol{\mathcal{A}} \cdot \boldsymbol{\mathcal{A}}^{-1} = \boldsymbol{\mathcal{A}}^{-1} \cdot \boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{I}}, \tag{6}$$

where $\boldsymbol{\mathcal{I}}$ represents the identity tensor of the same dimensions as $\boldsymbol{\mathcal{A}}$.

Tensor rank: The rank of a tensor, denoted as rank($\boldsymbol{\mathcal{A}}$), is a fundamental measure of its complexity. It represents the minimum number of basis tensors required to express $\boldsymbol{\mathcal{A}}$. Mathematically, it is written as

$$\text{rank}(\boldsymbol{\mathcal{A}}) = \min\{r : \boldsymbol{\mathcal{A}} = \sum_{i=1}^{r} \boldsymbol{\mathcal{U}}_i\}, \tag{7}$$

where $\boldsymbol{\mathcal{U}}_i$ are basis tensors and r is the rank.

III. Problem Formulation

A tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times \dots n_d}$ with mode d and size $n_1 \times n_2 \times \cdots \times n_d$ can be used to approximate the discretization of a multivariate function $f(x_1, x_2, \dots, x_d)$, where each entry of the tensor $\boldsymbol{\mathcal{X}}_{i_1, i_2, \dots, i_d}(i_1, i_2, \dots, i_d \in \mathbb{N})$ corresponds to the grid-based sampling of $f(i_1, i_2, \dots, i_d)$.

A d-mode tensor is rank one if it can be written as the outer product of d vectors, i.e.,

$$\boldsymbol{\mathcal{X}} = \boldsymbol{a_1} \circ \boldsymbol{a_2} \circ \cdots \circ \boldsymbol{a_d}, \tag{8}$$

which can be seen as discretization of a separable function $f(x_1, x_2, \dots, x_d) = f_1(x_1) f_2(x_2) \dots f_d(x_d)$, where each vector $\boldsymbol{a_i}$ with length n_i corresponds to a discretization of f_i (the *basis function* [27]) within a certain range. A more compact form using the Kronecker product can be written as

$$\boldsymbol{\mathcal{X}} = \otimes_{i=1}^{d} \boldsymbol{a_i}. \tag{9}$$

The CP decomposition factorizes a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times \ldots n_d}$ as a sum of rank one tensors, i.e.,

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} \boldsymbol{a_{r,1}} \circ \boldsymbol{a_{r,2}} \circ \cdots \circ \boldsymbol{a_{r,d}}, \tag{10}$$

or

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} \otimes_{i=1}^{d} \boldsymbol{a_i}, \tag{11}$$

where the *tensor rank* is defined as the smallest number of rank-one tensors (smallest decomposition rank R) that generate $\boldsymbol{\mathcal{X}}$ as their sum. With CP decomposition, the size of a tensor is reduced from $n_1 \times n_2 \times \cdots \times n_d$ to $(n_1 + n_2 + \ldots n_d)R$, which scales linearly with both d and R and becomes computationally attractive with a small R. However, in contrast to matrices, finding the rank of tensors has been proven to be generally NP-hard [23]. On the other hand, with dimension d often determined by the problem in hand, a small R becomes essential for further reducing the complexity. Hence, we seek to address the low-rank tensor approximation problem, formulated as

$$\begin{aligned} \min_{\hat{\boldsymbol{\mathcal{X}}}} \quad & ||\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}||^2 + \alpha R \\ \text{s.t.} \quad & \hat{\boldsymbol{\mathcal{X}}} = \sum_{r=1}^{R} \otimes_{i=1}^{d} \boldsymbol{a_i}, \end{aligned} \tag{12}$$

where $\boldsymbol{\mathcal{X}}$ is a given tensor, and α is the weighting factor used to balance the trade-off between approximation accuracy and decomposition rank.

The challenge of solving problem (12) roots in the implicit expression of the tensor rank R that is usually pre-given when computing the tensor approximation problem. According to the accuracy from the approximation results, adjustment on the rank value may be required after each calculation, where the rank value is increased or decreased by one for each adjustment [24], [26], [27]. For example, if a tensor approximation algorithm, e.g., ALS, stagnates before achieving the desired accuracy, a random rank-one tensor is then added to $\hat{\boldsymbol{\mathcal{X}}}$. The procedure is repeated until a rank that leads to the desired approximation accuracy is obtained. Several iterations may be required till an appropriate rank value is sought with satisfactory approximation accuracy, especially for problems with a high rank. This rank adaption process is time and resource-consuming. We aim to construct a formulation that explicitly represents the tensor rank such that the optimization problem formulated in (12) allows for minimizing the approximation error and rank value simultaneously. In the following, we first introduce a novel mixed-integer tensor representation, followed by the description of an adaptive low-rank tensor approximation algorithm.

IV. Adaptive Low-Rank Tensor Approximation

A. Mixed-Integer Tensor Representation

we introduce a binary variable set $k_r \in \{0, 1\}$ $(r = 1, \ldots, R)$ associated with each rank-one component in the CP decomposition,

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} k_r \cdot \boldsymbol{a_{r,1}} \circ \boldsymbol{a_{r,2}} \circ \cdots \circ \boldsymbol{a_{r,d}}, \tag{13}$$

where the binary constraint on k_r can be written as $k_r(k_r - 1) = 0$, for $r = 1, \ldots, R$. This mixed-integer representation allows us to select which rank-one tensor elements should be included in the approximation. According to the new representation, the reformulated problem is written as

$$\begin{aligned} \min_{\hat{\boldsymbol{\mathcal{X}}}} \quad & ||\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}||^2 + \alpha \sum_{r=1}^{R} k_r \\ \text{s.t.} \quad & \hat{\boldsymbol{\mathcal{X}}} = \sum_{r=1}^{R} k_r \cdot \boldsymbol{a_{r,1}} \circ \boldsymbol{a_{r,2}} \circ \cdots \circ \boldsymbol{a_{r,d}}, \\ & k_r(k_r - 1) = 0, \text{ for } r = 1, \ldots, R, \end{aligned} \tag{14}$$

where R is a sufficiently large integer we can assign to the unknown tensor. The mixed-integer tensor representation allows determining the rank-one tensor components as well as the tensor rank, denoted by the summation of the binary variables, simultaneously when solving the low-rank tensor approximation problem. To solve this challenging nonlinear optimization problem with both binary and continuous variables involved in a tensor representation, we propose an ALS and truncation combined approach. For clarity, we will first introduce the traditional ALS method [23] used for tensor approximation with a given rank.

B. Alternating Least Square (ALS) Method

The ALS method is a widely adopted approach for computing the CP decomposition of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times \ldots n_d}$ with a given rank R, written as

$$\begin{aligned} \min_{\hat{\boldsymbol{\mathcal{X}}}} \quad & ||\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}||^2 \\ \text{s.t.} \quad & \hat{\boldsymbol{\mathcal{X}}} = \sum_{r=1}^{R} \lambda_r \boldsymbol{a_{r,1}} \circ \boldsymbol{a_{r,2}} \circ \cdots \circ \boldsymbol{a_{r,d}}, \end{aligned} \tag{15}$$

where λ_r is the coefficient obtained by normalizing vectors $\boldsymbol{a_{r,1}}, \boldsymbol{a_{r,2}}, \ldots, \boldsymbol{a_{r,d}}$.

A minimum of $||\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}||^2$ satisfies $\nabla_{\boldsymbol{a}} ||\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}||^2 = 0$, where $\nabla_{\boldsymbol{a}}$ denotes the gradient with respect to all vector elements $\boldsymbol{a_{r,i}}(n_j)$ for $r = 1, \ldots, R$, $i = 1, \ldots, d$, and $n_j \in \{n_1, \ldots, n_d\}$. However, $\nabla_{\boldsymbol{a}} ||\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}||^2$ is highly nonlinear with respect to the vector elements $\boldsymbol{a_{r,i}}(n_j)$. ALS method resolves this issue by solving one dimension at a time while fixing the others.

By collecting all vectors of the same dimension in all rank-one components, the CP decomposition of a tensor can also be represented as Khatri-Rao product of the *factor matrix* $A_i = [\boldsymbol{a_{i,1}}, \boldsymbol{a_{i,2}}, \ldots, \boldsymbol{a_{i,R}}] \in \mathbb{R}^{n_i \times R}$, expressed as

$$\boldsymbol{\mathcal{X}}_{(i)} = A_i \Lambda (A_d \odot A_{d-1} \odot \ldots A_{i+1} \odot A_{i-1} \odot \cdots \odot A_1)^T, \tag{16}$$

where $\boldsymbol{\mathcal{X}}_{(i)}$ represents the mode-i matrixization of tensor $\boldsymbol{\mathcal{X}}$, Λ is a diagonal matrix formed by normalization factor λ_r and $\odot$ is Khatri-Rao product.

We consider an example of a 3rd-order tensor for better illustration while noting that the extension of (16) to higher-order cases is straightforward. With factor matrices, (15) can be written as

$$\min_{\hat{A_1}, \hat{A_2}, \hat{A_3}} \quad ||\boldsymbol{\mathcal{X}}_{(1)} - \hat{A_1} \Lambda (\hat{A_2} \odot \hat{A_3})^T||^2. \tag{17}$$

The ALS method first fixes $\hat{A_2}$ and $\hat{A_3}$, then problem (17) becomes a linear problem that is to optimize the least square error, i.e.,

$$\min_{\hat{A_1}} \quad ||\boldsymbol{\mathcal{X}}_{(1)} - \hat{A_1} \Lambda (A_3 \odot A_2)^T||^2, \tag{18}$$

which leads to the optimal solution

$$\hat{A_1}^* = \boldsymbol{\mathcal{X}}_{(1)} [(A_3 \odot A_2)^T]^{-1}. \tag{19}$$

With the properties of Khatri-Rao product, the solution can be rewritten as

$$\hat{A_1}^* = \boldsymbol{\mathcal{X}}_{(1)} (A_3 \odot A_2)(A_3^T A_3 * A_2^T A_2)^{-1}, \tag{20}$$

where $*$ is the Hadamard product (elementwise product) of matrices. In this form, the pseudoinverse of an $R \times R$ matrix rather than an $n_1 n_2 \times R$ matrix is calculated.

At each iteration for $i = 1, \ldots, d$, the ALS method solves for the optimal factor matrix in the corresponding dimension while keeping the other dimensions fixed, and updates $\hat{\boldsymbol{\mathcal{X}}}_i$ by (16). The procedure repeats until (a) the average point residual $\frac{||\hat{\boldsymbol{\mathcal{X}}} - \boldsymbol{\mathcal{X}}||^2}{\prod_{i=1}^{d} n_i}$ is lower than a prescribed approximation error tolerance ϵ; or (b) the improvement of the objective function is smaller than a prescribed threshold; or (c) a predefined maximum number of iterations T is reached. The full ALS procedure for a general d-order tensor approximation is shown in Algorithm 1.

Algorithm 1: ALS Method for CP Decomposition with Given Rank

Data: Tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times \ldots n_d}$, decomposition rank R
Result: Approximation of $\boldsymbol{\mathcal{X}}$ with rank R in factor matrices format (Λ, A_i for $i = 1, \ldots, d$) of CP decomposition

Initialize factor matrices $A_i \in \mathbb{R}^{n_i \times R}$ *for* $i = 1, \ldots, d$.;
Define residual tolerance ϵ.;
repeat
  for $i = 1, \ldots, d$ **do**
    $V \leftarrow (A_1^T A_1 * \cdots * A_{i-1}^T A_{i-1} * A_{i+1}^T A_{i+1} * \cdots * A_d^T A_d)$;
    $A_i \leftarrow \boldsymbol{\mathcal{X}}(A_d \odot \cdots \odot A_{i+1} \odot A_{i-1 \odot \cdots \odot A_1}) V^{-1}$;
    foreach *column vector* $\boldsymbol{a_r}$ *of* A_i **do** $\lambda_r \leftarrow$ norm($\boldsymbol{a_r}$);
    `// normalize columns of` A_i
  end
  $\Lambda \leftarrow$diag(λ_r);
  $\hat{\boldsymbol{\mathcal{X}}}_{(1)} \leftarrow A_1 \Lambda (A_d \odot \ldots A_2)^T$;
until $\frac{||\hat{\boldsymbol{\mathcal{X}}}_{(1)} - \boldsymbol{\mathcal{X}}_{(1)}||^2}{\prod_{i=1}^{d} n_i} < \epsilon$;
return Λ, A_i *for* $i = 1, \ldots, d$;

Although the ALS method is originally applied to computing CP decomposition with a given rank R, it is straightforward to extend it to a low-rank tensor approximation problem [24], [26], [27]. By starting with a low-rank value, ALS is repeatedly used to obtain an updated approximation result when a random rank-one tensor is added to the tensor obtained in the last loop. The procedure is repeated until the rank is high enough such that a good approximation (average point residual $< \epsilon$) can be obtained. More details of the iterative ALS (iALS) method can be referred to [26]. As we described above, the iALS process for low-rank tensor approximation is time and resource-consuming. We now propose a new algorithm to solve the mix-integer low-rank tensor approximation problem formulated in (14).

C. Integrated ALS and Truncation Method for Mixed-Integer Tensor Approximation

Following the tensor matrixization operation in (16), we first transform problem (14) into a matrix formulation through the factor matrices. For illustration simplicity, we consider the 3rd-order case again, written as

$$\begin{aligned} \min_{A_1, A_2, A_3, K} \quad & ||\boldsymbol{\mathcal{X}}_{(1)} - A_1 K (A_3 \odot A_2)^T||^2 + \alpha ||K||^2 \\ \text{s.t.} \quad & K = \text{diag}(k_r), \\ & k_r(k_r - 1) = 0, \text{ for } r = 1, \ldots, R, \end{aligned} \tag{21}$$

where $K \in \mathbb{R}^{R \times R}$ is a diagonal matrix with k_r, $r = 1, \ldots, R$, as its diagonal entries.

To solve this challenging nonlinear problem with both binary and continuous variables involved in $\boldsymbol{\mathcal{X}}$, we propose a two-stage optimization framework based on integrated ALS and truncation method. First, we relax the binary variables k_r as continuous variables. Then, problem (21) becomes

$$\begin{aligned} \min_{A_1, A_2, A_3, K} \quad & ||\boldsymbol{\mathcal{X}}_{(1)} - A_1 K (A_3 \odot A_2)^T||^2 + \alpha ||K||^2 \\ \text{s.t.} \quad & K = \text{diag}(k_r). \end{aligned} \tag{22}$$

In the first stage, we adopt the ALS algorithm to solve problem (22) with all binary variables k_r initialized as 1. By fixing matrix K, problem (22) is a CP decomposition problem with a given rank, akin to problem (15). Leveraging the ALS algorithm, in the inner iteration, we take alterations for each optimal factor matrix A_i while fixing the others. The optimal solution for the first-factor matrix is given by

$$\hat{A_1}^* = \boldsymbol{\mathcal{X}}_{(1)} (A_3 \odot A_2)(A_3^T A_3 * A_2^T A_2)^{-1} K^{-1}. \tag{23}$$

After each iteration of ALS, i.e., each factor matrix has been updated, we fix all factor matrices A_i and then start the second stage to update K. The problem in the second stage can be written as

$$\begin{aligned} \min_{K} \quad & ||\boldsymbol{\mathcal{X}}_{(1)} - A_1 K (A_3 \odot A_2)^T||^2 + \alpha ||K||^2 \\ \text{s.t.} \quad & K = \text{diag}(k_r), \end{aligned} \tag{24}$$

or in the Kronecker product form

$$\min_{k_r} \quad ||\boldsymbol{\mathcal{X}} - \sum_{r=1}^{R} k_r \cdot \otimes_{i=1}^{3} \boldsymbol{a_{i,r}}||^2 + \alpha \sum_{r=1}^{R} k_r^2. \tag{25}$$

According to the first-order optimality condition, we have $\nabla_{k_r} L = 0$, where ∇_{k_r} denotes the gradient with respect to all k_r, $r = 1, \ldots, R$, and L denotes the objective function in problem (25). Then for each individual k_r, the optimal solution satisfies

$$\nabla_{k_r} L = k_r || \otimes_{i=1}^{3} \boldsymbol{a_{i,r}}||^2 + ||(\hat{\boldsymbol{\mathcal{X}}}_r - \boldsymbol{\mathcal{X}}) * \otimes_{i=1}^{3} \boldsymbol{a_{i,r}}|| + \alpha k_r = 0, \tag{26}$$

where

$$\hat{\boldsymbol{\mathcal{X}}}_r = \sum_{\substack{q=1 \\ q \neq r}}^{R} k_q \cdot \otimes_{i=1}^{3} \boldsymbol{a_{i,q}}. \tag{27}$$

After obtaining each k_r by solving the linear equation (26), we proceed to a truncation step in order to reduce the decomposition rank. In this step, we introduce a threshold, ϵ_t, that is chosen to be a small value and serves as the truncation criterion. For $k_r \leq \epsilon_t$, we neglect the corresponding rank-one component. Consequently, the corresponding columns of the factor matrices are also eliminated, which reduces the computational cost for the ALS algorithm in the first stage. The two stages are iterated until the objective function reaches a threshold ϵ_s such that $||\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}||^2 \leq \epsilon_s$.

This two-stage optimization framework, aiming at solving the original tensor approximation problem (15) while penalizing k_r, allows us to gradually approach a low-rank CP decomposition of a given tensor. Achieving a balance between approximation accuracy and the decomposition rank is crucial, and we accomplish this equilibrium by introducing a weighting factor denoted as α. The value of α is related to tensor size ($n_1 \times n_2 \times \cdots \times n_d$), which reflects the trade-off between average point approximation error and the value of decomposition rank. The entire framework of the proposed algorithm for a general d-mode tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times \ldots n_d}$ is outlined in Algorithm 2.

Algorithm 2: Adaptive Low Rank Tensor Approximation

Data: Tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times \ldots n_d}$, sufficiently large R
Result: Low-rank approximation of $\boldsymbol{\mathcal{X}}$ in factor matrices format (K, A_i for $i = 1, \ldots, d$) of CP decomposition

Initialize factor matrices $A_i \in \mathbb{R}^{n_i \times R}$ *for* $i = 1, \ldots, d$; $K \leftarrow I^{R \times R}$
Define weighting factor α, *truncation criteria* ϵ_t *and stop criteria* ϵ_s.
repeat
  for $i = 1, \ldots, d$ **do**
    $V \leftarrow (A_1^T A_1 * \cdots * A_{i-1}^T A_{i-1} * A_{i+1}^T A_{i+1} * \cdots * A_d^T A_d)$;
    $A_i \leftarrow \boldsymbol{\mathcal{X}}(A_d \odot \cdots \odot A_{i+1} \odot A_{i-1 \odot \cdots \odot A_1}) V^{-1} K^{-1}$;
  end
  for $r = 1, \ldots, R$ **do**
    $\hat{\boldsymbol{\mathcal{X}}}_r \leftarrow \sum_{\substack{q=1 \\ q \neq r}}^{R} k_q \cdot \otimes_{i=1}^{d} \boldsymbol{a_{i,q}}$;
    $k_r \leftarrow \frac{||(\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}_r) * \otimes_{i=1}^{d} \boldsymbol{a_{i,r}}||}{\alpha + ||\otimes_{i=1}^{d} \boldsymbol{a_{i,r}}||^2}$;
  end
  foreach $k_r < \epsilon_t$ **do** $k_r \leftarrow 0$ and eliminate corresponding columns in A_i for $i = 1, \ldots, d$;
  $K \leftarrow$diag(k_r);
until *Improvement of objective function* $< \epsilon_s$;
return K, A_i *for* $i = 1, \ldots, d$;

V. Simulation Examples

To demonstrate the advanced computational performance of the proposed algorithm, we perform several numerical simulations and compare the results with the iALS [26]. We also evaluate how different prescribed parameters affect the performance of our algorithm. All simulations are performed on an AMD Ryzen 9 5900HS CPU. The codes are implemented with Python and the package *Tensorly* is used for basic calculation of tensor.

A. Comparison with iALS method

In the first experiment, we compare our method with the iALS method in terms of computation time, approximation accuracy, and decomposition rank. For the tensor to approximate, we randomly generate d factor matrices with size $N \times R$. The tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N \times N \times \ldots N}$ with d mode and rank R is then computed by these random factor matrices. We applied the two algorithms to approximate the resulting tensor and evaluate their performance with different values of size N and rank R. As for algorithm parameters, we set the weighting factor $\alpha = 10^{-2}$, truncation criteria $\epsilon_t = 10^{-3}$ and stop criteria $\epsilon_s = 10^{-3}$. It is worth noting that, due to normalization, we multiply α with the tensor size N^d when the penalty term is introduced to the objective function, which can be equivalently written as

$$\min_{\hat{\boldsymbol{\mathcal{X}}}} \quad \frac{||\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}||^2}{\prod_{i=1}^{d} n_i} + \alpha R, \tag{28}$$

where the first term represents the average point approximation error. For the initial approximate rank, we set it to be 1 for iALS and $2R$ for our algorithm, so that both algorithms have to increase (or decrease) R from the initial rank to the ideal one. The above setting of algorithm parameters is adopted throughout all simulation cases.

For the first case, we set $d = 3$, $N = 200$ and change the rank R of the tensor to be approximated to evaluate two algorithms with different tensor rank values. In this ideal setting, since all target tensors are generated without introducing any error, both methods converge to the actual rank with small approximation errors ($< 10^{-3}$). However, due to the efficiency of the mixed-integer tensor representation and proposed algorithm, our method demonstrates the advantage in computation time with increasing R, shown in Fig. 1(a). In the second case, we set $d = 3$, $R = 20$, and N variate, the time efficiency advantage of our proposed method becomes more significant with a large N. While both algorithms converge to the actual rank with small approximation errors, our proposed method uses much less computation time, especially for a large N, as shown in Fig. 1(b). From the results of these two cases, our proposed method is capable of efficiently computing the low-rank approximation of given tensors, especially for large-size problems with a high inherent CP rank.

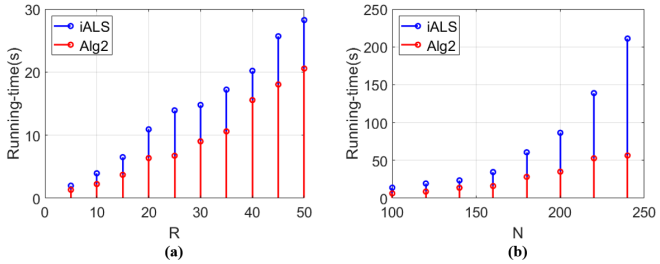

Fig. 1: Computation time of iALS and Algorithm 2. (a) Dimension $d = 3$, size $N = 200$ with variate designated rank R; (b) Dimension $d = 3$, designated rank $R = 20$ with variate size N.

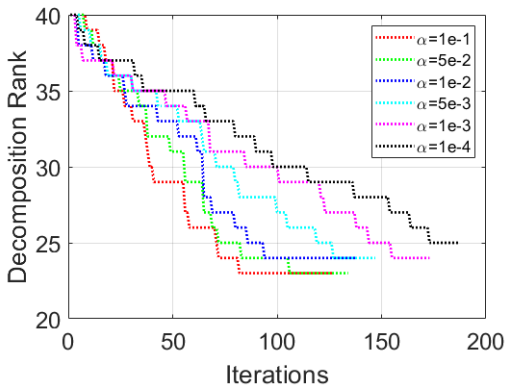

Fig. 2: Convergence of decomposition rank with different choices of weighting factor α

B. Evaluation of weighting factor α

In this set of simulation cases, the tensor to be approximated is computed by randomly generated 3 factor matrices of size (200×20), resulting $d = 3$, $N = 200$, $R = 20$. However, different from the ideal case, a Gaussian noise of relative magnitude 5% is introduced to the generated tensor. In this setting, an approximation with a higher decomposition rank may lead to improved approximation accuracy, while a higher rank also implies a low data compression rate. Hence, balancing the trade-off between approximation error and decomposition rank becomes important. Our method makes it feasible to achieve the trade-off by adjusting

α	0.1	0.05	0.01	0.005	0.001	10^{-4}
Error	0.0432	0.0367	0.0083	0.0079	0.0031	0.0015
Rank	23	23	24	24	24	25

TABLE I: Approximation error and decomposition rank with different weighting factor α

the value of the weighting factor α. Note that α is multiplied by the tensor size to normalize the objective function (28). As a result, a reasonable choice for α should be on the same order of magnitude as the relative fitting error required for the application problem. To evaluate the effect of the weighting factor, we examine the convergence of decomposition rank and the relative approximation error ($||\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}||/||\boldsymbol{\mathcal{X}}||$) with different choices of α. The results are shown in Fig. 2 and Table I. It demonstrates that a larger weighting factor α can lead to lower decomposition rank, but it also results in lower approximation accuracy.

VI. Conclusion

In this paper, we propose an adaptive low-rank tensor approximation algorithm based on mixed-integer representations to efficiently solve the CP decomposition problem of a given tensor with a balance between high approximation accuracy and a low-rank value. To achieve this, we introduce the extra penalty term of decomposition rank and transform the low-rank approximation problem into a mixed-integer nonlinear optimization problem. We then develop an alternating optimization framework to iteratively solve for tensor approximation while minimizing the decomposition rank. Through the combination of alternating least square method and truncation approach, the computational cost is reduced and the convergence rate is enhanced, especially for large-scale problems. Besides, by adjusting the weighting factor of the penalty term, our proposed algorithm is able to balance the trade-off between approximation error and rank, which validates the effectiveness and versatility of our approach for a wide range of tensor decomposition tasks. Furthermore, the simulation results demonstrate the superiority of our method in terms of both accuracy and efficiency when compared to existing state-of-the-art approaches.

References

[1] H. Zhou, L. Li, and H. Zhu, "Tensor regression with applications in neuroimaging data analysis," *Journal of the American Statistical Association*, vol. 108, no. 502, pp. 540–552, 2013.

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

[3] V. Rajagopalan, Z. Jiang, J. Stojanovic-Radic, G. Yue, E. P. Pioro, G. Wylie, and A. Das, "A basic introduction to diffusion tensor imaging mathematics and image processing steps," *Brain Disord Ther*, vol. 6, no. 229, p. 2, 2017.

[4] M. Mørup, "Applications of tensor (multiway array) factorizations and decompositions in data mining," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, 2011.

[5] B. N. Khoromskij, "Tensor numerical methods for high-dimensional pdes: Basic theory and initial applications," *arXiv: Numerical Analysis*, 2014.

[6] L. Grasedyck, D. Kressner, and C. Tobler, "A literature survey of low-rank tensor approximation techniques," *GAMM-Mitteilungen*, vol. 36, no. 1, pp. 53–78, 2013.

[7] R. A. Harshman, "Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-model factor analysis," 1970.

[8] J. D. Carroll and J. J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition," *Psychometrika*, vol. 35, pp. 283–319, 1970.

[9] G. Beylkin and M. Mohlenkamp, "Algorithms for numerical analysis in high dimensions," *SIAM J. Scientific Computing*, vol. 26, pp. 2133–2159, 2005.

[10] G. Beylkin and M. J. Mohlenkamp, "Numerical operator calculus in higher dimensions," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 16, pp. 10246–10251, 2002.

[11] H. Liu, L. Daniel, and N. Wong, "Model reduction and simulation of nonlinear circuits via tensor decomposition," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 7, pp. 1059–1069, 2015.

[12] J. Deng, H. Liu, K. Batselier, Y.-K. Kwok, and N. Wong, "Storm: A nonlinear model order reduction method via symmetric tensor decomposition," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 557–562, 2016.

[13] Z. Li, Y. Jiang, and K. Xu, "Non-linear model-order reduction based on tensor decomposition and matrix product," *IET Control Theory & Applications*, vol. 12, no. 16, pp. 2253–2262, 2018.

[14] J. Yang, Y.-L. Jiang, and K.-L. Xu, "Nonlinear model order reduction with low rank tensor approximation," *Asian Journal of Control*, vol. 23, no. 1, pp. 255–264, 2021.

[15] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, pp. 279–311, 1966.

[16] Y. Chen and Y. Zhou, "Total variation regularized low-rank tensor approximation for color image denoising," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2523–2527, 2018.

[17] I. V. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[18] A. Dektor, A. Rodgers, and D. Venturi, "Rank-adaptive tensor methods for high-dimensional nonlinear pdes," *Journal of Scientific Computing*, vol. 88, no. 2, p. 36, 2021.

[19] L. Richter, L. Sallandt, and N. Nüsken, "Solving high-dimensional parabolic pdes using the tensor train format," in *Proceedings of the 38th International Conference on Machine Learning*, vol. 139, pp. 8998–9009, PMLR, 2021.

[20] V. Murg, F. Verstraete, R. Schneider, P. R. Nagy, and Ö. Legeza, "Tree tensor network state with variable tensor order: An efficient multireference method for strongly correlated systems," *Journal of Chemical Theory and Computation*, vol. 11, pp. 1027 – 1036, 2015.

[21] N. Nakatani and G. K.-L. Chan, "Efficient tree tensor network states (TTNS) for quantum chemistry: Generalizations of the density matrix renormalization group algorithm," *The Journal of Chemical Physics*, vol. 138, no. 13, p. 134113, 2013.

[22] J. Ballani, L. Grasedyck, and M. Kluge, "Black box approximation of tensors in hierarchical tucker format," *Linear Algebra and its Applications*, vol. 438, no. 2, pp. 639–657, 2013.

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

[24] Y. Sun and M. Kumar, "Solution of high dimensional transient fokker-planck equations by tensor decomposition," in *2015 American Control Conference (ACC)*, pp. 1475–1480, 2015.

[25] A. M. Boelens, D. Venturi, and D. M. Tartakovsky, "Parallel tensor methods for high-dimensional linear pdes," *Journal of Computational Physics*, vol. 375, pp. 519–539, 2018.

[26] M. B. Horowitz, A. Damle, and J. W. Burdick, "Linear hamilton jacobi bellman equations in high dimensions," in *53rd IEEE Conference on Decision and Control (CDC)*, pp. 5880–5887, 2014.

[27] E. Stefansson and Y. P. Leong, "Sequential alternating least squares for solving high dimensional linear hamilton-jacobi-bellman equation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3757–3764, 2016.

[28] S. Weiland and F. Van Belzen, "Singular value decompositions and low rank approximations of tensors," *IEEE transactions on signal processing*, vol. 58, no. 3, pp. 1171–1182, 2009.

[29] J. Ballani and L. Grasedyck, "A projection method to solve linear systems in tensor format," *Numerical linear algebra with applications*, vol. 20, no. 1, pp. 27–43, 2013.

[30] D. Kressner, M. Plešinger, and C. Tobler, "A preconditioned low-rank cg method for parameter-dependent lyapunov matrix equations," *Numerical Linear Algebra with Applications*, vol. 21, no. 5, pp. 666–684, 2014.