An Incremental Tensor-Train Decomposition for Cyber-Physical-Social Big Data

Huazhong Liu, Laurence T. Yang, *Senior Member, IEEE*, Yimu Guo, Xia Xie, and Jianhua Ma, *Member, IEEE*

Abstract— Cyber-physical-social big data generated from ubiquitous devices and diverse spaces generally are multi-source, heterogeneous, and deeply intertwined. To efficiently analyze and handle the ubiquitous cyber-physical-social big data, tensor is considered as an effective tool, but the curse of dimensionality is still the main bottleneck of tensor-based big data analysis. Tensor networks can considerably alleviate or overcome it through the tensor approximate theory. Therefore, this paper focuses on developing an efficient big data processing framework based on tensor networks and providing an incremental tensor train decomposition approach for the streaming big data. Concretely, this paper first presents a hierarchical cyber-physical-social big data processing framework composed of three planes, namely, data representation and decomposition, data storage and processing, and data analysis and service. Then, tensor train (TT) and quantized TT decompositions are particularly introduced to remarkably overcome the curse of dimensionality. Furthermore, to efficiently handle the continuous streaming big data and avoid the repeated decomposition for the history data, an incremental tensor train decomposition (ITTD) approach is proposed and the complexities are further analyzed in detail. Experimental results demonstrate that ITTD demonstrably outperforms the nonincremental TT decomposition in execution time on the precise of guaranteeing the nearly equal approximation error.

Index Terms—Cyber-physical-social systems, big data, ubiquitous computing, tensor decomposition, tensor network, incremental tensor train decomposition, quantized tensor train.

1 Introduction

With the continuous expansion of ubiquitous sensors, devices, and social media, all sorts of data generated from diverse spaces (e.g., cyber space, physical space, social space) become widely available and intertwined. Cyber-Physical-Social Systems (CPSS), an emerging paradigm, has received considerable attention in the academia and research community [1]. Differing from Cyber-Physical Systems (CPS) which merely focuses on connecting the physical world objects, CPSS integrates the information generated from not only cyber space and physical space, but also social space and further connects them into a large-scale, heterogeneous, federated, and complex system [2]. CPSS focuses on integrating and coordinating all data collected from these spaces to extract useful information and provide intelligent services for our lives. CPSS has spurred intensive popularity and been applied to multiple realms, such as context-aware vehicular cyberphysical systems [3], NextMe prediction systems [4], personalized travel sequence recommendation systems [5], to name a few.

With the pervasiveness of data acquisition systems and the booming development of social networks and communication networks, the data acquired from CPSS become ubiquitous and very huge in amount. Cyber-physical-social big data are gradually formed and exhibit some specific characteristics, such as multi-source, heterogeneous, real-time streaming, etc. [6]. Due to the rapid increase in data scale, traditional ubiquitous computing methods show their limitations, and the data analysis methods in CPSS involving data representation, storage, and processing require new approaches. Meanwhile, the cyber-physical-social big data are generated in a streaming way, it is necessary to implement an incremental big data processing solution to satisfy the real-time requirements in real-world. Therefore, how to efficiently represent, analyze, and handle the streaming cyber-physical-social big data faces numerous challenges.

Tensor, a big data analysis tool, has been adopted in diverse branches because of its prominent advantages in representing and handling the complex and high-order data, such as economy, education, agriculture, transport, and military [7]. Some tensor-based data analysis methods, such as tensor decompositions, tensor networks, high-dimensional clustering, and multi-factor prediction, have also been utilized to deal with some practical problems. Especially for some large scale scenarios appeared in stochastic partial differential equations [8], brain data analysis [9], large-scale network anomography [10], etc., where the tensor's order

- *H. Liu is with the School of Computer Science and Technology and Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the School of Information Science and Technology, Jiujiang University, Jiujiang 332005, China. E-mail: sharpshark_ding@163.com.*
- *L.T. Yang is with the School of Computer Science and Technology and Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the Department of Computer Science, St. Francis Xavier University, Antigonish, NS B2G 2W5, Canada. E-mail: ltyang@gmail.com.*
- *Y. Guo and X. Xie are with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: kknightgo@gmail.com, shelicy@hust.edu.cn.*
- *J. Ma is with the Faculty of Computer and Information Sciences, Hosei University, Tokyo 184-8584, Japan. E-mail: jianhua@hosei.ac.jp.*
- *(Corresponding author: Laurence T. Yang.)*

*Manuscript received Sep. **, 2017; revised ** **, ****.*

may reach up to 10, 100, even 1000. However, curse of dimensionality is still the main bottleneck during the tensor-based big data analysis. The curse of dimensionality refers to the fact that the entry number of a tensor grows exponentially with its order [11]. Given a tensor with the order $N = 50$ and the dimensionality $I = 2$, then the number of entries is $I^N = 2^{50}$. Suppose that each entry occupies 8 bytes in memory, then the volume of the tensor will reach up to $8PB$. Accordingly, the additional overhead including the amount of operations, computations, and running memory grow exponentially with the tensor's order.

In order to tackle these problems arising from such huge volume, the theories of tensor approximation and multilinear algebra play important roles in numerical analysis and practical applications. Tensor approximation provide us an approach and opportunity to resolve these problems via a small number of parameters. Therefore, some typical tensor decompositions (TDs), such as, CANDECOMP-PARAFAC (CP) decomposition, Tucker decomposition, etc., are presented successively and widely used to analyze tensor data [12]. However, they appear discommodious because of their corresponding limitations when dealing with high-order tensors. For CP decomposition, the decomposition algorithms generally are not stable for high-order tensors and the computation of the optimal rank is considered as an NP-hard problem [13]. For Tucker decomposition, although the decomposition algorithms are stable, it is also not suitable for high-order tensors in that the number of parameters of their core tensor is exponential with the tensor's order which remains suffered by the curse of dimensionality [14].

Recently, some tensor networks (TNs) are proposed to overcome these limitations when dealing with high-order tensors, such as Hierarchical Tucker (HT), Tensor Train (TT), Quantized Tensor Train (QTT) decompositions, etc. HT decomposition is considered as a multilevel variant of Tucker decomposition, it is also an alternative way to efficiently reduce the complexity of the Tucker decomposition [15], [16]. TT decomposition decomposes a high-order tensor into a series of low-order core tensors (typically 2nd-order or 3rd-order) which are interconnected by tensor contractions and connected to a linear train [11]. QTT decomposition decomposes a tensor with very high dimensionality as a series of sparsely interconnected low-order and low-dimensionality core tensors via synthetically exploiting quantization (or tensorization) and suitable TT decomposition [17].

TT, a special case of HT, is viewed as the simplest tensor network format and has some advantages. Firstly, TT decomposition is based on the low-rank approximation of auxiliary unfolding matrix, the decomposition algorithm is non-recursive and stable when facing high-order tensors. Secondly, the number of parameters of TT decomposition is asymptotically equal to that of CP decomposition [18], thus it does not suffer from the morass of curse of dimensionality. Thirdly, it is also feasible to directly implement some algebra operations in TT format, e.g., addition, dot product, norm, and matrix-by-vector, etc. [11], which are conducive to considerably improve the computation efficiency. Therefore, TT decomposition has already been successfully used to solve intractable high-dimensional problems in scientific computing, such as high-dimensional elliptic equations [19], extreme eigenvalues in higher dimensions [20], high-dimensional operator equations and eigenvalue [21], etc.

Nevertheless, facing the large-scale and heterogeneous cyber-physical-social big data, there remains many challenges during the tensor-based big data analysis process in practical scenarios. Firstly, the constructed tensor for representing the cyber-physical-social big data is hardly stored and handled explicitly because of the curse of dimensionality. Hence, it is required to select a suitable tensor decomposition approach to tackle the problem. Secondly, after the cyber-physical-social tensor is decomposed, how to execute some tensor operations based on the decomposition results is also a challenge. These tensor operations are hardly performed based on the original tensor in that they can not be reconstructed because of the curse of dimensionality. Therefore, it is very meaningful to promote a novel processing framework to efficiently represent, store, analyze, and handle the cyber-physical-social big data.

Besides, the cyber-physical-social big data are dynamic and generally generated in a streaming way. After the new streaming data are appended to the original tensor, how to compute the TT format of the updated tensor faces challenges. Firstly, it is impossible to explicitly reconstruct the decomposed tensor and further achieve the updated tensor because of the curse of dimensionality. Secondly, it is necessary to reduce the total decomposition time to satisfy the time constraint in practical ubiquitous scenarios. In the past research, some incremental analysis methods are successively presented, e.g., SVD-based incremental approaches [22], [23] and incremental tensor-based approaches [24], [25]. However, there are little relevant methods for the incremental tensor train decomposition. Hence, it is significant that how to reuse the original TT decomposition result of the history tensor and compute the new TT format of the updated tensor.

Therefore, this paper focuses on developing an efficient big data processing framework based on tensor networks and providing an incremental tensor train decomposition approach for streaming big data. This paper first develops a novel processing framework for cyber-physical-social big data by exploiting the advantages of tensor networks to realize efficient storage and analysis. The proposed three-layer processing framework supports data representation and decomposition, data storage and processing, and data analysis and service. Afterwards, to avoid the repeated computation for the history data and compute the new TT format of the updated tensor, this paper provides an incremental TT decomposition (ITTD) approach. Finally, a series of experiments are conducted to validate the proposed framework and ITTD approach.

To summarize, the major contributions of this paper are listed as follows.

- Presents a cyber-physical-social big data processing framework based on tensor networks to considerably alleviate or overcome the curse of dimensionality by exploiting their characteristics and advantages in decomposition format and algebraic operation.
- Proposes an ITTD approach to avoid the repeated computation for the history data and reduce the total execution time of decomposition when facing the continuous streaming big data.

- Experimental results demonstrate that the proposed ITTD approach demonstrably outperforms the non-incremental TT decomposition in reducing the total execution time on the precise of guaranteeing the nearly equal approximation error and storage space.

The rest of this paper is organized as follows. Section 2 briefly recalls the related preliminaries. Section 3 presents a novel processing framework based on tensor networks for cyber-physical-social big data. In section 4, an incremental tensor train decomposition approach is proposed in detail. Section 5 compares the experimental results and section 6 concludes this paper.

2 Preliminaries and Tensor Train Decomposition

Tensor, as the generalization of matrix in high-dimensional space, has been considered as an effective data representation and analysis tool for big data and has gained increasing attention by many researchers. For the concrete description about tensor operations, tensor decompositions, and tensor applications, please refer to [12], [18], [25], [26]. This section briefly reviews the presentation method based on tensor network diagram and tensor train decomposition.

2.1 Tensor Network Diagram

To visualize the corresponding operations and complex interactions between different tensors, tensor network diagram (TND) is graphically utilized [18]. Fig. 1 depicts the graphical representation of basic symbols and tensor operations through TND. In TND, there are generally two types of symbols, one is a series of nodes or shapers (e.g., squares, circles, spheres, ellipses, etc.) to graphically represent tensors, and the other is outgoing edges (or lines, branches, leads) emerging from one node to represent an order (or way, mode) of the tensor. The edges of TND are divided into two categories: connected edges which connect two nodes and represent a contraction of two respective tensors along the corresponding pair of orders, and free edges which connect to only one node and stand for a physical order of the tensor. Consequently, the order number of a tensor is decided by the number of free edges in TND. TND not only graphically provides illustrative large distributed networks but also performs complex tensor operations in an intuitive way and without using explicitly mathematical expressions.

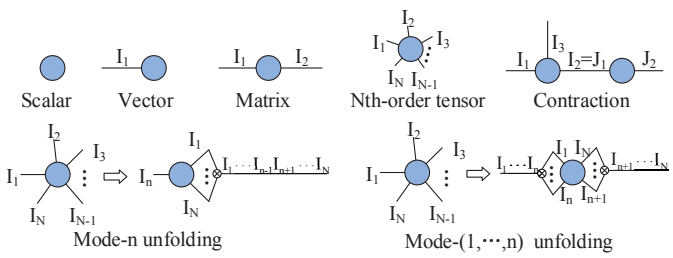


Fig. 1. Graphical representation of basic symbols and operations via TND.

2.2 Tensor Train Decomposition

Tensor train decomposition (TTD) is firstly proposed by Oseledets in the numerical analysis community [11]. In practice, it is similar with the concept of "matrix product states" [27], [28]. Formally, an Nth-order tensor $\underline{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is defined to be the TT-format if it satisfies the following format:

$$\underline{X} = \underline{X}^{(1)} \bullet \underline{X}^{(2)} \bullet \cdots \bullet \underline{X}^{(n)} \bullet \cdots \bullet \underline{X}^{(N)}, \quad (1)$$

where $\bullet$ denotes the operation of contracted product, $\underline{X}^{(n)} \in \mathbb{R}^{r_{n-1} \times I_n \times r_n}$ $(n = 1, \cdots, N;\ r_0 = r_N = 1)$ is referred to as core tensors (or cores), and $\{r_0, r_1, \cdots, r_N\}$ are viewed as TT-ranks of tensor. Fig. 2 illustrates the graphical representation of an Nth-order tensor in TT format. Alternatively, if the tensor is assigned to a specified index $(i_1, i_2, \cdots, i_N)$, then the entry-wise TT format can be represented as follows:

$$\underline{X}(i_1, i_2, \cdots, i_N) = \underline{X}^{(1)}(i_1)\underline{X}^{(2)}(i_2) \cdots \underline{X}^{(N)}(i_N), \quad (2)$$

where $\underline{X}^{(n)}(i_n) \in \mathbb{R}^{r_{n-1} \times r_n}(\ n = 1, \cdots, N;\ r_0 = r_N = 1)$.

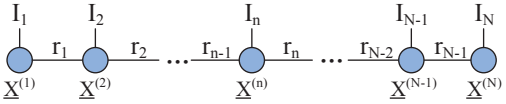


Fig. 2. Graphical representation of an Nth-order tensor in TT format.

According to [11], the implementation of TT decomposition is entirely based on low-rank decomposition through a series of SVDs of auxiliary unfolding matrices. The TT decomposition of an Nth-order tensor needs $N-1$ sequential SVDs. For each SVD on the auxiliary mode-$(1, \cdots, n)$ unfolding matrix, the left singular matrix is transformed as the current core tensor, the remainder will be continuously executed by SVD in the same manner, until the last core tensor is yielded. Because the low-rank tensor rarely appears in practical computation, we should compress the original data tensor by exploiting the low-rank approximation theory. At this point the exact low-rank decomposition is modified to the approximate case. While executing the SVD for each given unfolding matrix M_k, if the singular values are truncated at δ, i.e., $\left\| M_k - \widehat{M}_k \right\|_F \leq \delta \left\| M_k \right\|_F$ ($\widehat{M}_k$ is the reconstructed approximate matrix according to the truncated value), then the approximation error between the original Nth-order tensor $\underline{X}$ and the approximate tensor $\widehat{\underline{X}}$ will be $\varepsilon = \sqrt{N-1}\delta$, i.e., $\left\| \underline{X} - \widehat{\underline{X}} \right\|_F \leq \varepsilon \left\| \underline{X} \right\|_F$. For the proof, please refer to [11]. In other words, given any prescribed accuracy ε for the TT decomposition, the threshold of truncation for each unfolding matrix should be set to $\delta = \frac{\varepsilon}{\sqrt{N-1}}$. In this case, the TT-ranks are actually δ-ranks of the unfolding matrices.

Consequently, according to the above decomposition process and error estimation, the tensor train decomposition algorithm can be provided for any Nth-order tensor with any prescribed accuracy ε, whose pseudocode is described as Algorithm 1 [11].

3 Cyber-physical-social Big Data Processing Framework Based on Tensor Networks

In this section, a cyber-physical-social big data processing framework based on tensor networks is illustrated, the

Algorithm 1: Tensor Train Decomposition Algorithm

Input: An Nth-order tensor $\underline{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a prescribed accuracy ε.
Output: TT cores $\underline{X}^{(1)}, \underline{X}^{(2)}, \cdots, \underline{X}^{(N)}$ of approximation tensor $\widehat{\underline{X}}$ satisfying $\left\|\underline{X} - \widehat{\underline{X}}\right\|_F \leq \varepsilon \left\|\underline{X}\right\|_F$.
begin
  /* Initialize variables */
  $\delta \leftarrow \frac{\varepsilon}{\sqrt{N-1}}$; $T \leftarrow \underline{X}$; $r_0 \leftarrow 1$;
  for $n \leftarrow 1$ *to* $N-1$ **do**
    /* Mode-$[n]$ unfolding */
    $M \leftarrow$ **Unfolding-n** $(T, [r_{n-1}I_n, \frac{numel(M)}{r_{n-1}I_n}])$;
    /* Compute δ-truncated SVD */
    $[U, S, V] \leftarrow$ **SVD** (M, δ);
    i.e., $M = USV^T + E$, $\|E\|_F \leq \delta \|M\|_F$;
    /* Achieve new TT-rank and core */
    $r_n \leftarrow rank_\delta(M)$;
    $\underline{X}^{(n)} \leftarrow$ **Tensorization**$(U, [r_{n-1}, I_n, r_n])$;
    /* New temporary tensor */
    $T \leftarrow SV^T$;
  /* Achieve last core and TT-rank */
  $\underline{X}^{(N)} \leftarrow$ **Tensorization**$(T, [r_{N-1}, I_N, r_N])$;
  $r_N \leftarrow 1$;
  return cores $\underline{X}^{(1)}, \underline{X}^{(2)}, \cdots, \underline{X}^{(N)}$.

overview is depicted in Fig. 3. The hierarchical processing framework consists of three planes, namely, data representation and decomposition plane, data storage and processing plane, and data analysis and service plane. We elaborate the respective functions and responsibilities of each plane from a bottom-up view.

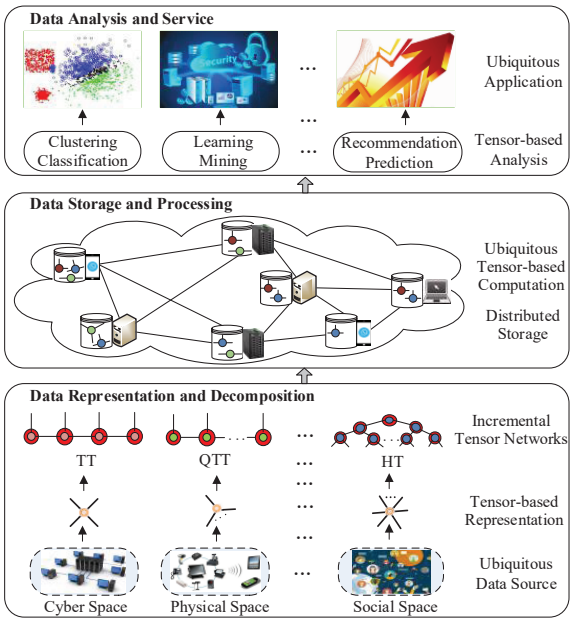


Fig. 3. Cyber-physical-social big data processing framework based on tensor networks.

3.1 Data Representation and Decomposition

Data representation and decomposition plane aims to represent the sensed and collected data from CPSS in a compact and efficient way. Ubiquitous cyber-physical-social big data derived from diverse spaces generally are multi-source and heterogeneous, hence, tensor is applied to this framework by virtue of its excellent characteristics in low-rank approximation and high-dimensional correlative analysis. Building on the previous work of our team in [25], the structured, semi-structured, and unstructured data with different types (such as text, image, audio, video, etc.) can be represented to low-order tensors. In general, the original constructed tensor not only takes up tremendous storage space, but also contains a great deal of noisy and redundant data. To reduce the storage space and extract the significant implicit features from the original tensor, tensor networks (TNs) can be applied under different practical situations, such as TT, QTT, HT, etc. The achieved data after implementing TNs generally become higher in quality and smaller in volume, which are conductive to the compact storage and accurate calculation, as well as the effective communication.

Besides, some data tensors derived from CPSS are characterised by very large dimensionality in some actual situations, such as very large-scale vectors or matrices. On that occasion, quantized tensor train (QTT) decomposition can be exploited. The implementation of QTT generally can be achieved from the following two steps. Firstly, we reformat the original tensor with high dimensionality to a higher-order tensor with low dimensionality via tensorization or quantization. Secondly, we apply a suitable TN (e.g., TT) to decompose the tensorized tensor. Fig. 4 provides two examples of QTT decomposition. Such a quantized tensor can achieve a good compression performance if it admits low-rank approximation [29]. Moreover, to deal with the streaming data and meet the requirements of response time in CPSS, some incremental updating methods should be exploited to avoid the repeated computation for the history data and reduce the computation time.

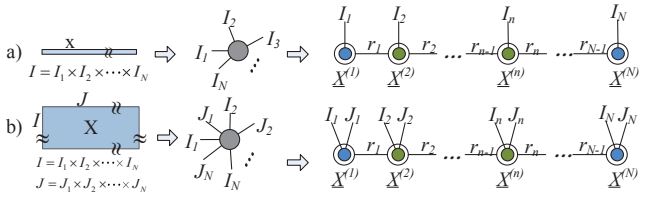


Fig. 4. Examples of quantized tensor-train decomposition.

3.2 Data Storage and Processing

Data storage and processing plane is responsible to realize the distributed data storage and parallel tensor computation based on the decomposed result through ubiquitous computing. Differing from the standard TDs (e.g., CP or HOSVD) which decompose a tensor to only one core tensor, TNs aim to decompose a high-order tensor into a sequence of low-order core tensors (typically 2nd-order or 3rd-order). These cores become relatively smaller in size and are interconnected by some specific contracted orders, such as the TT decomposition of an Nth-order tensor in Fig. 2.

TNs have different decomposed formats, but they can be reciprocally converted from one to the other [30]. This kind of decomposition approaches can considerably alleviate or overcome the adverse effects caused by the curse of dimensionality in computation and storage (please see the analysis in section 4.5). Therefore, these cores can be stored in data centers or clouds or ubiquitous computing devices in a distributed manner only if we grasp the relationship among cores according to their specific rules.

In addition, some TN formats have some special properties and support some algebraic and tensor-based operations, such as addition, minus, matrix-by-vector, and norm in TT format [11]. These operations can be directly computed in the same TN format without reconstructing these cores to the original tensors, and the result will also maintain the same TN format. Therefore, these original algebraic and other tensor-based operations can be converted to the operations among these distributed low-order cores, including typical tensor addition, Hadamard product, inner product, multilinear product, Einstein product, etc. They can be feasibly executed in parallel under ubiquitous computing environment. It is beneficial to further improves the efficiency of tensor networks based big data analysis.

3.3 Data Analysis and Service

Data analysis and service plane focuses on providing effective and proactive services for people based on the analysis results according to the practical ubiquitous intelligence scenarios. The core tensors after executing TNs generally contain enormous latent feature information. Based on these cores, several big data analysis methods, such as high-dimensional alternative clustering algorithms, multi-dimensional correlative analysis algorithms, personalized recommendation and accurate prediction algorithms, can be performed to mine valuable knowledge hidden in cyber-physical-social big data. Owing to the integration of heterogeneous data derived from multiple spaces and the multi-dimensional correlative analysis, their analysis results commonly are more targeted and accurate. These results are beneficial for providing excellent services for people according to ubiquitous intelligence scenarios and concrete application requirements, such as proactive healthcare services in smart monitor systems, accurate recommendations for learners in smart educational systems, and accurate prediction of traffic jams in smart transport systems, etc.

4 Incremental Tensor-Train Decomposition

Generally, the data in CPSS are generated in a streaming manner, to avoid the repeated computation for the history data, this section detailedly illustrates an incremental TT decomposition (ITTD) approach. The proposed incremental approach is performed by virtue of the special mathematical properties of TT format rather than the incremental SVD.

4.1 Main Solution

Suppose that the original tensor is $\underline{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_k \times \cdots \times I_N}$, and the incremental tensor along the kth order is $\underline{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_k^{'} \times \cdots \times I_N}$, hence, the updated tensor is $\underline{Z} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times (I_k + I_k^{'}) \times \cdots \times I_N}$. The main process of ITTD is described as Fig. 5, which is comprised of four steps. (1) Execute the TT decomposition of the incremental tensor $\underline{Y}$ and yield its TT format $\underline{Y}_{tt}$. It is noted that the TT format $\underline{X}_{tt}$ of original tensor $\underline{X}$ is known and has been stored in advance. (2) Compute their TT cores of the original and incremental zero-padding tensors on the basis of the decomposed TT cores $\underline{X}_{tt}$ and $\underline{Y}_{tt}$, represented as $\underline{X}_{tt}^{'}$ and $\underline{Y}_{tt}^{'}$, respectively. Thereinto, appending zeros to the original and incremental tensors is to maintain the consistency of their dimensionality. (3) Compute the TT format $\underline{Z}_{tt}^{'}$ of the updated tensor $\underline{Z}$ directly by exploiting the addition of their TT cores of the original and incremental zero-padding tensors. (4) Orthogonalize and compress the TT cores of $\underline{Z}_{tt}^{'}$, which can be periodically implemented in actual situations. That is because the new TT cores after executing the addition do not necessarily maintain the orthogonality. Hereto, the ultimate TT format $\underline{Z}_{tt}$ of updated tensor $\underline{Z}$ is yielded.

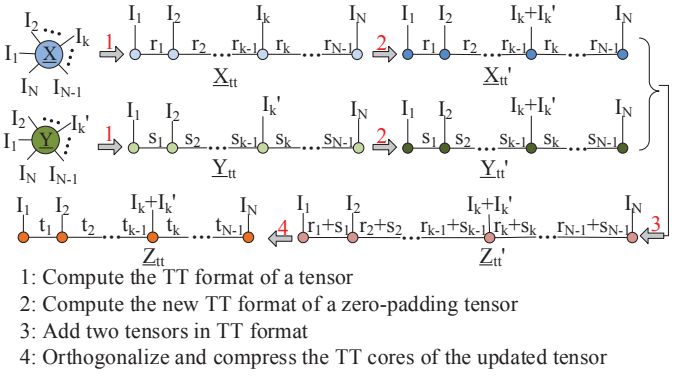


Fig. 5. Main process of the incremental tensor train decomposition.

For a more detailed explanation, Fig. 6 illustrates a specific example of the incremental tensor train decomposition. Given a 3rd-order tensor $\underline{X} \in \mathbb{R}^{2\times3\times3}$, suppose that the incremental tensor is $\underline{Y} \in \mathbb{R}^{1\times3\times3}$, which will be appended to $\underline{X}$ along the first order. To compute the TT format of the updated tensor $\underline{X}+\underline{Y}$, we first compute the TT format of the incremental tensor $\underline{Y}$ and represent it as $\underline{Y}_{tt}$, and the TT format of the original tensor $\underline{X}$ can be directly achieved from the historical results and depicted as $\underline{X}_{tt}$. Secondly, we append zeros to the original and incremental tensors to maintain the consistency of their dimensionality, and compute the TT format ($\underline{X}_{tt}^{'}$ and $\underline{Y}_{tt}^{'}$) of the zero-padding tensors on the basis of the existing TT cores ($\underline{X}_{tt}$ and $\underline{Y}_{tt}$). Thirdly, we directly compute the sum of two zero-padding tensors in TT format. Finally, we orthogonalize the TT cores of the updated tensor to maintain their orthogonality and compress them according to the prescribed accuracy.

4.2 TT Format of Zero-padding Tensor

Suppose that the TT format of a given tensor is known, if we append enough zeros to the given tensor along a specified order, how can we compute the new TT format of the zero-padding tensor based on the known TT cores of the given tensor? This section is to provide an effective solution to compute the TT format for the zero-padding tensor.

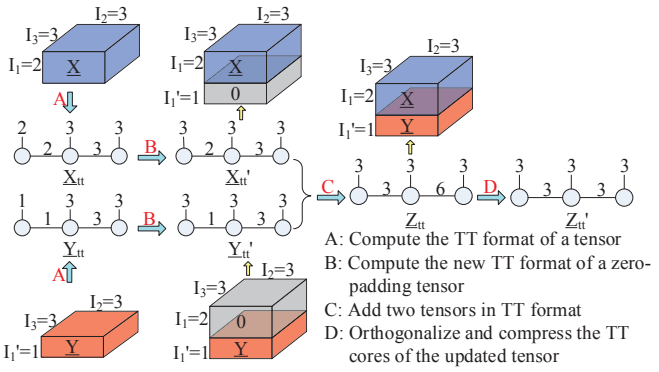


Fig. 6. Example of the incremental tensor train decomposition for a 3rd-order tensor along the first order.

Before providing the computation approach, two theorems are illustrated and proved in detail, they are extended and improved on the basis of the conclusion in [22].

Theorem 1. *Given a matrix $M_1 \in \mathbb{R}^{m\times n}$, and a matrix $M_2 \in \mathbb{R}^{(m+\Delta m)\times n}$ which is constructed by appending zeros on the bottom rows of M_1, i.e., $M_2 = \begin{bmatrix} M_1 \\ 0 \end{bmatrix}$. Supposed that the SVDs of M_1 and M_2 are represented as $M_1 = U_1 S_1 V_1^T$, $M_2 = U_2 S_2 V_2^T$. If the same δ-truncations are applied in the SVDs of M_1 and M_2, and their δ-ranks are r_1 and r_2, then $r_2 = r_1$, $U_2^{r_2} = \begin{bmatrix} U_1^{r_1} \\ 0 \end{bmatrix}$, and $V_2^{r_2} = V_1^{r_1}$.*

Proof. *According to the properties of SVD , we can know that U_1, V_1, U_2, and V_2 are orthogonal matrices, S_1 and S_2 are diagonal matrices. Therefore,*

$$M_1^T M_1 = (V_1 S_1^T U_1^T)(U_1 S_1 V_1^T) = V_1 {S_1}^2 V_1^T, \tag{3}$$

$$M_2^T M_2 = (V_2 S_2^T U_2^T)(U_2 S_2 V_2^T) = V_2 {S_2}^2 V_2^T. \tag{4}$$

Consider $M_2^T M_2 = \begin{bmatrix} M_1^T & 0 \end{bmatrix} \begin{bmatrix} M_1 \\ 0 \end{bmatrix} = M_1^T M_1$, by integrating Eq. 3 and Eq. 4, we can obtain $V_1 {S_1}^2 V_1^T = V_2 {S_2}^2 V_2^T$. It is noted that S_1^2 and S_2^2 are the eigenvalue matrices of the two equal symmetric matrices $M_1^T M_1$ and $M_2^T M_2$, respectively. According to the uniqueness of eigenvalues, it can be inferred that S_1^2 and S_2^2 are equal, thus their corresponding singular values of M_1 and M_2 are also equal. Accordingly, it can be easily concluded that V_1 and V_2 are equivalent, i.e., $V_2 = V_1$. If the same δ-truncation are applied in the SVDs of M_1 and M_2, then their δ-ranks shall be equal, i.e., $r_2 = r_1$. Furthermore, we can infer that $V_2^{r_2} = V_1^{r_1}$.

Besides, suppose that $U_2 = \begin{bmatrix} U_1 \\ 0 \end{bmatrix}$, according to the above analysis, we can know that S_1 and S_2 are equal, V_1 and V_2 are equivalent. Therefore, $M_2 = U_2 S_2 V_2^T = \begin{bmatrix} U_1 \\ 0 \end{bmatrix} S_2 V_2^T = \begin{bmatrix} U_1 S_2 V_2^T \\ 0 \end{bmatrix} = \begin{bmatrix} U_1 S_1 V_1^T \\ 0 \end{bmatrix} = \begin{bmatrix} M_1 \\ 0 \end{bmatrix}$. It shows that the assumption is correct. Therefore, we can further obtain that $U_2^{r_2} = \begin{bmatrix} U_1^{r_1} \\ 0 \end{bmatrix}$.

Theorem 2. *Given a matrix $N_1 = [v_1, v_2, \cdots, v_n] \in \mathbb{R}^{m\times n}$, and let $N_2 = [v_1, 0, v_2, \cdots, 0, \cdots, 0, \cdots, v_n] \in \mathbb{R}^{m\times(n+\Delta n)}$ in which Δn zero column vectors are inserted to N_1. Suppose that the SVDs of N_1 and N_2 are expressed as $N_1 = P_1 S_1 Q_1^T$, $N_2 = P_2 S_2 Q_2^T$. If the same δ-truncations are applied in the SVDs of N_1 and N_2, and their δ-ranks are r_1 and r_2, then $r_2 = r_1$, $P_2^{r_2} = P_1^{r_1}$, and $Q_2^{r_2} = \begin{bmatrix} Q_1^{r_1} \\ 0 \end{bmatrix}$.*

Proof. *Clearly, by employing some elementary matrix transformations, $N_2 = [v_1, 0, v_2, \cdots, 0, \cdots, 0, \cdots, v_n]$ can be transformed to $N_2 = [v_1, v_2, \cdots, v_n, 0, \cdots, 0] = [N_1, 0]$. If we transpose the matrix M_2 in Theorem 1, then ${M_2}^T = \begin{bmatrix} {M_1}^T, 0 \end{bmatrix}$. We can see that their forms of the two equations are identical. Therefore, according to the conclusion in Theorem 1, it can be easily infer that $r_2 = r_1$, $P_2^{r_2} = P_1^{r_1}$, $Q_2^{r_2} = \begin{bmatrix} Q_1^{r_1} \\ 0 \end{bmatrix}$.*

Suppose that there is a given Nth-order tensor $\underline{X} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_k\times\cdots\times I_N}$. If we append zeros to $\underline{X}$ along the kth order, the zero-padding tensor can be generated and represented as $\underline{X}^{'} \in \mathbb{R}^{I_1\times I_2\times\cdots\times(I_k+I_k^{'})\times\cdots\times I_N}$. $\underline{X}^{'}$ contains the given tensor (i.e., subtensor $\underline{X}^{'}(:,:,\cdots,(1:I_k),\cdots,:)$) and the appended zero tensor (i.e., subtensor $\underline{X}^{'}(:,:,\cdots,(I_k+1:I_k+I_k^{'}),\cdots,:)$), which is exhibited in Fig. 6. Suppose that the given tensor has already been decomposed to TT format and stored in advance, which is described as follows:

$$\begin{aligned} &\underline{X}(i_1, i_2, \cdots, i_k, \cdots, i_N) \\ &= \underline{X}^{(1)}(i_1)\underline{X}^{(2)}(i_2)\cdots\underline{X}^{(k)}(i_k)\cdots\underline{X}^{(N)}(i_N), \end{aligned} \tag{5}$$

where $\underline{X}^{(n)}(i_n) \in \mathbb{R}^{r_{n-1}\times r_n}(n = 1, 2, \cdots, N; r_0 = r_N = 1)$.

To achieve the TT format of zero-padding tensor from the existing TT format of the given tensor, we illustrate the TT decomposition process via tensor network diagram. Fig. 7 graphically demonstrates the TT decomposition process of an Nth-order tensor, where $N-1$ SVDs are executed successively. For illustrating the decomposition process in more depth, the implementation details of the second step (i.e., step (2) in Fig. 7) are interpreted as follows. Firstly, we unfold the tensor derived from the previous step (i.e., box (T1) in Fig. 7) to a matrix (i.e., box (M) in Fig. 7) according to mode-$(1,\cdots,n)$ unfolding operation. Then SVD ($M = USV^T$) is executed on the unfolding matrix, the left singular matrix (i.e., box (U) in Fig. 7) is reshaped to the current core tensor $\underline{X}^{(2)}$, and the remaining section (i.e., box (SV^T) in Fig. 7) will be viewed as the input of next step (i.e., box (T2) in Fig. 7) and decomposed continuously.

Therefore, while computing the first $k-1$ TT cores of the zero-padding tensor $\underline{X}^{'}$, the kth order is always located in the column of each unfolding matrix. After unfolding tensors $\underline{X}$ and $\underline{X}^{'}$ along the first order, we can find that the unfolding matrix $M_1^{'}$ generated from tensor $\underline{X}^{'}$ can be yielded by inserting some zero column vectors in the unfolding matrix M_1 generated from tensor $\underline{X}$, namely, $M_1^{'} = \begin{bmatrix} M_1 & 0 \end{bmatrix}$, which is illustrated in Fig. 8(a). If we decompose the two matrices through δ-SVD and express them as $M_1^{'} \approx U_1^{r_1'} S_1^{r_1'} {V_1^{r_1'}}^T$ and $M_1 \approx U_1^{r_1} S_1^{r_1} {V_1^{r_1}}^T$. According to Theorem 2, it can be inferred that their left singular matrices are equivalent, i.e., ${U_1^{r_1}}^{'} = U_1^{r_1}$, and their right singular matrices shall satisfy: ${V_1^{r_1}}^{'} = \begin{bmatrix} V_1^{r_1} \\ 0 \end{bmatrix}$. In accordance with the foregoing analysis in Fig. 7, the left singular matrix

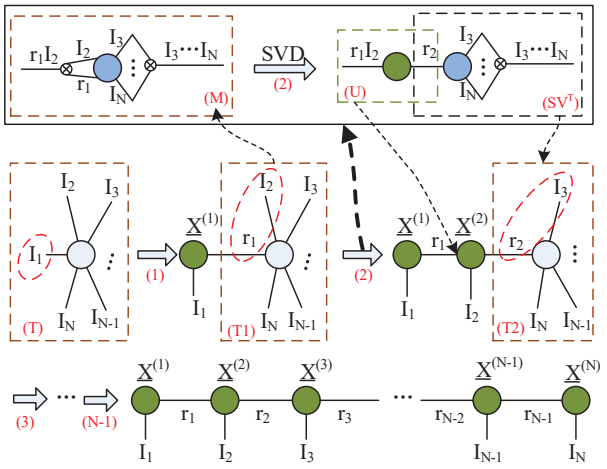


Fig. 7. Illustration of tensor train decomposition for an Nth-order tensor.

will be transformed as the first core tensor, it means that the first TT cores of tensors $\underline{X}$ and $\underline{X}^{'}$ are equal. And the remaining section $S_1^{r_1'}V_1^{r_1'T} = \begin{bmatrix} S_1^{r_1}V_1^{r_1T} & 0 \end{bmatrix}$ will be transformed to the input matrix of the second step, and its form is same with the first unfolding matrix $M_1^{'}$. Similarly, the second input matrix will be decomposed in the same manner until the $(k-1)$th core tensor is yielded. Therefore, we can conclude that the first $k-1$ TT cores of tensor $\underline{X}^{'}$ and $\underline{X}$ are identical.

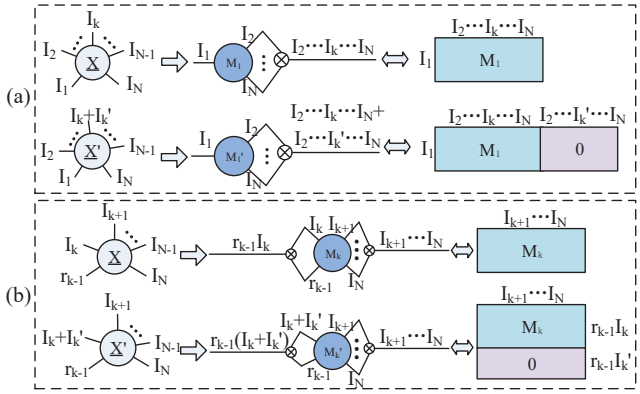


Fig. 8. Illustration of tensor unfolding for $\underline{X}$ and $\underline{X}^{'}$.

While computing the kth core of the zero-padding tensor $\underline{X}^{'}$, the kth order is just located in the row of the unfolding matrix $M_k^{'}$, and the bottom rows of $M_k^{'}$ are all zeros, i.e., $M_k^{'} = \begin{bmatrix} M_k \\ 0 \end{bmatrix}$, which is illustrated in Fig. 8(b). Then we execute δ-SVD for $M_k^{'}$ and M_k and express them as $M_k^{'} \approx U_k^{r_k'}S_k^{r_k'}V_k^{r_k'T}$ and $M_k \approx U_k^{r_k}S_k^{r_k}V_k^{r_kT}$. According to Theorem 1, it can be seen that $U_k^{r_k'} = \begin{bmatrix} U_k^{r_k} \\ 0 \end{bmatrix}$ and will be transformed as the kth TT core. Consequently, we can achieve the kth TT core of the zero-padding tensor $\underline{X}^{'}$ by appending zeros to the existing kth TT core of the given tensor $\underline{X}$ according to corresponding transformation rules. Besides, their right singular matrices $V_k^{'}$ and V_k are equivalent in accordance with the conclusion in Theorem 1. It illustrates that the following unfolding matrices maintain unchanged and the corresponding TT cores from $k+1$ to N accordingly are the same.

Therefore, all TT cores of the zero-padding tensor $\underline{X}^{'}$ can be achieved from the TT cores of the given tensor $\underline{X}$ without repeatedly recomputing the TT decomposition. According to aforementioned analysis, we can find that all TT cores remain unchanged except for the kth core. If the TT format of the zero-padding tensor $\underline{X}^{'}$ is represented as follows:

$$\begin{aligned} &\underline{X}^{'}(i_1, i_2, \cdots, i_n \cdots, i_N) \\ &= \underline{X}^{'(1)}(i_1)\underline{X}^{'(2)}(i_2)\cdots\underline{X}^{'(n)}(i_n)\cdots\underline{X}^{'(N)}(i_N), \end{aligned} \tag{6}$$

where $\underline{X}^{'(n)}(i_n) \in \mathbb{R}^{r_{n-1}^{'} \times r_n^{'}}$ $(n = 1, 2, \cdots, N)$. Then the rules of yielding all new TT cores $\underline{X}^{'(n)}(i_n)$ $(n = 1, 2, \cdots, N)$ of the zero-padding tensor $\underline{X}^{'}$ are as follows:

$$\underline{X}^{'(n)}(i_n) = \begin{cases} 0, & n = k \ and \ i_n > I_k; \\ \underline{X}^{(n)}(i_n), & otherwise. \end{cases} \tag{7}$$

Meanwhile, it can be inferred that the TT ranks remain unchanged after appending zeros to the matrix according to Theorems 1 and 2.

$$r_n^{'} = r_n, n = 1, 2, \cdots, N. \tag{8}$$

4.3 Addition of Two Tensors in TT Format

According to the approach provided in section 4.2, we can compute the TT format of the original and incremental zero-padding tensors. Then our current objective is how to compute the TT format of the updated tensor based on the TT formats of the original and incremental zero-padding tensors. Therefore, our solution is to directly merge their corresponding TT cores by exploiting the addition characteristic of TT format.

According to Eq. 7, the TT format of the original zero-padding tensor and incremental zero-padding tensor can be yielded and represented as follows:

$$\underline{X}^{'}(i_1, i_2, \cdots, i_N) = \underline{X}^{'(1)}(i_1)\underline{X}^{'(2)}(i_2)\cdots\underline{X}^{'(N)}(i_N), \tag{9}$$

where $X^{'(n)}(i_n) \in \mathbb{R}^{r_{n-1} \times r_n}$ $(n = 1, \cdots, N; r_0 = r_N = 1)$.

$$\underline{Y}^{'}(i_1, i_2, \cdots, i_N) = \underline{Y}^{'(1)}(i_1)\underline{Y}^{'(2)}(i_2)\cdots\underline{Y}^{'(N)}(i_N), \tag{10}$$

where $Y^{'(n)}(i_n) \in \mathbb{R}^{s_{n-1} \times s_n}$ $(n = 1, \cdots, N; s_0 = s_N = 1)$.

Suppose that the TT format of the updated tensor is represented as:

$$\underline{Z}^{'}(i_1, i_2, \cdots, i_N) = \underline{Z}^{'(1)}(i_1)\underline{Z}^{'(2)}(i_2)\cdots\underline{Z}^{'(N)}(i_N), \tag{11}$$

where $Z^{'(n)}(i_n) \in \mathbb{R}^{t_{n-1} \times t_n}$ $(n = 1, \cdots, N)$. Then the computation of the TT format of the updated tensor can be converted to the merge of original TT cores and incremental TT cores. According to the addition operation provided in [11], the generation rule of updated TT cores is defined as:

$$\underline{Z}^{'(n)}(i_n) = \begin{cases} \begin{pmatrix} \underline{X}^{'(1)}(i_1) & \underline{Y}^{'(1)}(i_1) \end{pmatrix}, & n = 1; \\ \begin{pmatrix} \underline{X}^{'(n)}(i_n) & 0 \\ 0 & \underline{Y}^{'(n)}(i_n) \end{pmatrix}, & n = 2, \cdots, N-1; \\ \begin{pmatrix} \underline{X}^{'(N)}(i_N) \\ \underline{Y}^{'(N)}(i_N) \end{pmatrix}, & n = N. \end{cases} \tag{12}$$

Actually, by substituting Eq. 12 into Eq. 11 and exploiting continuous matrix multiplication, we can obtain:

$$\begin{aligned}
&\underline{\mathcal{Z}}^{'}(i_1,i_2,\cdots,i_N)=\underline{\mathcal{Z}}^{'(1)}(i_1)\underline{\mathcal{Z}}^{'(2)}(i_2)\cdots\underline{\mathcal{Z}}^{'(N)}(i_N)\\
&=\begin{pmatrix}\underline{\mathcal{X}}^{'(1)}(i_1) & \underline{\mathcal{Y}}^{'(1)}(i_1)\end{pmatrix}\begin{pmatrix}\underline{\mathcal{X}}^{'(2)}(i_2) & 0\\ 0 & \underline{\mathcal{Y}}^{'(2)}(i_2)\end{pmatrix}\cdots\\
&\cdots\begin{pmatrix}\underline{\mathcal{X}}^{'(N-1)}(i_{N-1}) & 0\\ 0 & \underline{\mathcal{Y}}^{'(N-1)}(i_{N-1})\end{pmatrix}\begin{pmatrix}\underline{\mathcal{X}}^{'(N)}(i_N)\\ \underline{\mathcal{Y}}^{'(N)}(i_N)\end{pmatrix}\\
&=\underline{\mathcal{X}}^{'(1)}(i_1)\underline{\mathcal{X}}^{'(2)}(i_2)\cdots\underline{\mathcal{X}}^{'(N-1)}(i_{N-1})\underline{\mathcal{X}}^{'(N)}(i_N)\\
&+\underline{\mathcal{Y}}^{'(1)}(i_1)\underline{\mathcal{Y}}^{'(2)}(i_2)\cdots\underline{\mathcal{Y}}^{'(N-1)}(i_{N-1})\underline{\mathcal{Y}}^{'(N)}(i_N)\\
&=\underline{\mathcal{X}}^{'}(i_1,i_2,\cdots,i_N)+\underline{\mathcal{Y}}^{'}(i_1,i_2,\cdots,i_N).
\end{aligned}\tag{13}$$

From Eq. 13, it can be seen that any entry of tensor $\underline{\mathcal{Z}}^{'}$ is the sum of the corresponding entries of tensor $\underline{\mathcal{X}}^{'}$ and $\underline{\mathcal{Y}}^{'}$. Besides, it can be inferred from Eq. 12 that the TT ranks of the updated tensor $\underline{\mathcal{Z}}^{'}$ are the sum of the corresponding TT ranks of the original and incremental zero-padding tensors $\underline{\mathcal{X}}^{'}$ and $\underline{\mathcal{Y}}^{'}$, which is illustrated as follows:

$$t_n=\begin{cases}1, n=1 \text{ or } N;\\ r_n+s_n, n=2,3,\cdots,N-1.\end{cases}\tag{14}$$

4.4 Orthogonalization and Compression of Updated TT Cores

The TT format of the updated tensor can be efficiently computed through the aforementioned approach. Nonetheless, some new TT cores after executing the addition of two tensors in Eq. 12 do not necessarily satisfy the orthogonality. Meanwhile, their TT ranks may also be increased. Therefore, to maintain the orthogonalization of updated TT cores and avoid the growth of TT ranks while satisfying the prescribed accuracy, the orthogonalization and compression of updated TT cores can be selectively implemented. In practice, to save the execution time, it is not necessary to execute the orthogonalization and compression operation after every updating. Generally, the process should be executed in the following situations. (1) If the update times reach up to a setting threshold, it can be periodically executed offline. (2) If the storage space of cores reaches up to a warning value. (3) If the requirement of orthogonalization is required before some special operations.

Algorithm 2 gives the pseudocode of orthogonalizing and compressing the updated TT cores [11]. Lines $4-12$ are to orthogonalize TT cores whose size are $r_{n-1}\times I_n r_n$ through QR decomposition from right to left, where Q is orthogonal and will be transformed to the new orthogonal TT core and R is triangle matrix and will be passed to the left TT core. Lines $13-22$ are responsible for compressing TT cores whose size is $r_{n-1}I_n\times r_n$ through SVD operations from left to right while maintaining the prescribed accuracy.

Therefore, according to the solution in section 4.1, the ITTD algorithm can be realized according to the proposed approach, the pseudocode is illustrated in algorithm 3.

Algorithm 2: Algorithm for the Orthogonalization and Compression of Updated TT Cores

1 **Input**: TT cores $\underline{\mathcal{Z}}^{(1)},\underline{\mathcal{Z}}^{(2)},\cdots,\underline{\mathcal{Z}}^{(N)}$ of an Nth-order updated tensor $\underline{\mathcal{Z}}$ and a prescribed accuracy ε.
2 **Output**: Orthogonal TT cores $\underline{\mathcal{Z}}^{'(1)},\underline{\mathcal{Z}}^{'(2)},\cdots,\underline{\mathcal{Z}}^{'(N)}$ of the approximate tensor $\underline{\mathcal{Z}}^{'}$. The computed approximation satisfies $\left\|\underline{\mathcal{Z}}-\underline{\mathcal{Z}}^{'}\right\|_F\le\varepsilon\|\underline{\mathcal{Z}}\|_F$.
3 **begin**
4 /* Compute truncation parameters */
5 $\delta\leftarrow\frac{\varepsilon}{\sqrt{N-1}}$;
6 /* Right-to-left orthogonalization */
7 **for** $n\leftarrow N$ *to* 2 **do**
8 /* Reshape the current core */
9 $M_n\leftarrow$ **Reshape** ($\underline{\mathcal{Z}}^{(n)}$, $[r_{n-1},I_n*r_n]$);
10 /* Execute QR operation */
11 $[Q,R]\leftarrow$ **QR** (M_n);
12 $\underline{\mathcal{Z}}^{(n)}\leftarrow$ **Reshape** (Q,$[r^{'}_{n-1},I_n,r_n]$);
13 /* Change next core tensor */
14 $\underline{\mathcal{Z}}^{(n-1)}\leftarrow\underline{\mathcal{Z}}^{(n-1)}\times_3 R$;
15 **if** $\varepsilon\neq 0$ **then**
16 /* Left-to-right compression */
17 **for** $n\leftarrow 1$ *to* $N-1$ **do**
18 /* Reshape the current core */
19 $M_n\leftarrow$ **Reshape** ($\underline{\mathcal{Z}}^{(n)}$, $[r_{n-1}*I_n,r_n]$);
20 /* Execute SVD operation */
21 $[U,S,V]\leftarrow$ **SVD** (M,δ);
22 $\underline{\mathcal{Z}}^{'(n)}\leftarrow$ **Reshape** (U, $[r_{n-1},I_n,r^{'}_n]$);
23 /* Change next core tensor */
24 $\underline{\mathcal{Z}}^{(n+1)}\leftarrow\underline{\mathcal{Z}}^{(n+1)}\times_1 SV^T$;
25 return cores $\underline{\mathcal{Z}}^{'(1)},\underline{\mathcal{Z}}^{'(2)},\cdots,\underline{\mathcal{Z}}^{'(N)}$.

4.5 Complexity Analysis

4.5.1 Time Complexity Analysis

The execution time of the proposed ITTD approach consists of TT decomposition of incremental tensor, computation of new TT format for zero-padding tensors, addition of two tensors in TT format, and orthogonalization and compression of TT cores. Let T_{itt}, T_{ttzero}, T_{ttadd}, and T_{ttorth} denote the execution time consumed by each section. Therefore, the total time consumption T_{ITTD} can be expressed as:

$$T_{ITTD}=T_{itt}+T_{ttzero}+T_{ttadd}+T_{ttorth}.\tag{15}$$

Given an Nth-order tensor $\underline{\mathcal{X}}\in\mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, for simplicity, we assume that $I=max\{I_n\}$ $(n=1,\cdots,N)$ and $r=\{r_0,r_1,\cdots,r_N\}$, where r_n is the TT rank of the tensor. If the incremental dimensionality along the specific order is I' (in general, $I'<I$), then the maximal dimensionality of the original, incremental, and updated tensors are I, I, and $I+I'$, respectively.

To compute the TT decomposition of an incremental tensor, $N-1$ successive SVDs need to be executed and cost $\mathcal{O}(I^{N+1})\sim\mathcal{O}(I^{\frac{N}{2}+2}{r_{N/2-1}}^2)$ operations. The analysis process can be illustrated as follows:

Consider the first $N/2$ SVDs, we can calculate the consumed time according to the size of unfolding matrices.

$n=1, I*I^{N-1}, \mathcal{O}(I^{N-1}*I^2)=\mathcal{O}(I^{N+1})$;
$n=2, Ir_1*I^{N-2}, \mathcal{O}(I^{N-2}*(Ir_1)^2)=\mathcal{O}(I^N{r_1}^2)$;
$n=3, Ir_2*I^{N-3}, \mathcal{O}(I^{N-3}*(Ir_2)^2)=\mathcal{O}(I^{N-1}{r_2}^2)$;
.....

Algorithm 3: The Process of ITTD Algorithm

Input: The TT format $\underline{X}_{tt}$ of the original Nth-order tensor $\underline{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_k \times \cdots \times I_N}$, an incremental tensor $\underline{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_k^{'} \times \cdots \times I_N}$ and a prescribed accuracy ε. **Output**: TT cores $\underline{Z}^{(1)}, \underline{Z}^{(2)}, \cdots, \underline{Z}^{(N)}$ of the updated tensor $\underline{Z} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times (I_k + I_k^{'}) \times \cdots \times I_N}$ in TT format.

begin

`/* Compute the TT format of tensor` $\underline{Y}$ `according to algorithm 1 */`

$\underline{Y}_{tt}$ ($\underline{Y}^{(1)}, \underline{Y}^{(2)}, \cdots, \underline{Y}^{(N)}$) $\leftarrow$ **TT**($\underline{Y}$, ε);

`/* Compute TT format of zero-padding tensors according to Eq. 7 */`

$\underline{X}_{tt}^{'}$ ($\underline{X}^{'(1)}, \underline{X}^{'(2)}, \cdots, \underline{X}^{'(N)}$) $\leftarrow$ **TT_zeros**($\underline{X}_{tt}$, k);

$\underline{Y}_{tt}^{'}$ ($\underline{Y}^{'(1)}, \underline{Y}^{'(2)}, \cdots, \underline{Y}^{'(N)}$) $\leftarrow$ **TT_zeros**($\underline{Y}_{tt}$, k);

`/* Execute addition of two tensors in TT format according to Eq. 12 */`

$\underline{Z}_{tt}^{'}$ ($\underline{Z}^{'(1)}, \underline{Z}^{'(2)}, \cdots, \underline{Z}^{'(N)}$) $\leftarrow$ **Add_TT**($\underline{X}_{tt}^{'}$, $\underline{Y}_{tt}^{'}$);

if *the requirement of orthogonalization and compression is satisfied* **then**

`/* Orthogonalize and compress the TT cores of updated tensor acoording to algorithm 2 */`

$\underline{Z}^{(1)}, \underline{Z}^{(2)}, \cdots, \underline{Z}^{(N)} \leftarrow$ **Orth_TT**($\underline{Z}_{tt}^{'}$, ε);

return TT cores $\underline{Z}^{(1)}, \underline{Z}^{(2)}, \cdots, \underline{Z}^{(N)}$.

$$
\begin{aligned}
&n = \tfrac{N}{2}, Ir_{N/2-1} * I^{\frac{N}{2}}, \mathcal{O}(I^{\frac{N}{2}} * (Ir_{N/2-1})^2) \\
&= \mathcal{O}(I^{\frac{N}{2}+2} {r_{N/2-1}}^2); \\
&\cdots\cdots
\end{aligned}
$$

It can be inferred that the total decomposition time is less than twice the sum of the time consumed by the first $N/2$ SVDs. Therefore, $T_{itt} = \mathcal{O}(2 * (I^{N+1} + I^N {r_1}^2 + I^{N-1} {r_2}^2 + \cdots + I^{\frac{N}{2}+2} {r_{N/2-1}}^2))$. If $r_1 \leq \sqrt{I}, r_2 \leq \sqrt{I} r_1, r_3 \leq \sqrt{I} r_4, \cdots, r_{N/2-1} \leq \sqrt{I} r_{N/2-2}$, then $T_{itt} = \mathcal{O}(I^{N+1})$; and if $r_1 \geq \sqrt{I}, r_2 \geq \sqrt{I} r_1, r_3 \geq \sqrt{I} r_4, \cdots, r_{N/2-1} \geq \sqrt{I} r_{N/2-2}$, then $T_{itt} = \mathcal{O}(I^{\frac{N}{2}+2} {r_{N/2-1}}^2)$.

If all unfolding matrices are full rank and no truncation is applied to all SVDs, then the worst time complexity of TT decomposition will be generated and can be represented as $T_{itt} = \mathcal{O}(2 * (I^{N+1} + I^{N+2} + I^{N+3} + \cdots + I^{\frac{3N}{2}})) = \mathcal{O}(I^{\frac{3N}{2}})$. Suppose that $r = max\{r_n\}(n = 0, 1, \cdots, N)$, if $r \geq I^{\frac{N-2}{4}}$, the time complexity is $T_{itt} = \mathcal{O}(I^{\frac{N}{2}+2} r^2)$, and if r increases to $I^{\frac{N}{2}-1}$, it will reach up to the worst time complexity $\mathcal{O}(I^{\frac{3N}{2}})$.

According to Eq. 7, the computation of the new TT format for the zero-padding tensor need only to change the kth TT core, which just costs $T_{ttzero} = \mathcal{O}(Ir^2)$ operations. According to Eq. 12, the addition of two tensors in TT format costs $T_{ttadd} = \mathcal{O}(NIr^2)$ operations. In the process of orthogonalization and compression of TT cores, the right-to-left orthogonalization requires successive $N-1$ QRs for matrices of size $2r$-by-$2rI$ and costs $\mathcal{O}(N * 2rI * (2r)^3) = \mathcal{O}(16NIr^4)$ operations, the left-to-right compression requires $N-1$ SVDs for matrices of size $2rI$-by-$2r$ and costs $\mathcal{O}(16NIr^4)$ operations. Hence, the total time of orthogonalization and compression of TT cores is $T_{ttorth} = \mathcal{O}(32NIr^4)$. Therefore, according to Eq. 15, the total time consumption is:

$$
T_{ITTD} = \begin{cases} O(I^{N-1} I^{'2} + NIr^4), r \leq I^{\frac{N-2}{4}} \\ O(I^{\frac{N}{2}+2} r^2 + NIr^4), r > I^{\frac{N-2}{4}}. \end{cases} \tag{16}
$$

Besides, the time complexity of nonincremental TT decomposition (NTTD) is:

$$
T_{NTTD} = \begin{cases} O((I + I^{'})^{N+1}), r^{'} \leq (I + I^{'})^{\frac{N-2}{4}} \\ O((I + I^{'})^{\frac{N}{2}+2} r^{'2}), r^{'} > (I + I^{'})^{\frac{N-2}{4}}. \end{cases} \tag{17}
$$

To compare other incremental HOSVD (IHOSVD) in [25], its time complexity is illustrated as follows:

$$
T_{IHOSVD} = O(N(I^{N-1} r^2 + I^N I^{'})). \tag{18}
$$

From the time complexity analysis in Eq. 16, we can find that T_{ITTD} is mainly consumed by the TT decomposition of incremental tensor and the orthogonalization and compression of TT cores. If N and I are larger or r is smaller, then the orthogonalization and compression procedure will account for less proportion of execution time. By comparing Eqs. 16 and 17, we can find that $r \leq r^{'}$. Therefore, after analyzing Eqs. 16, 17 and 18, it can be inferred that ITTD takes greater advantage in execution time compared with NTTD and IHOSVD when the order and dimensionality of tensor are larger and the compression rank is smaller, which is conducive to the cyber-physical-social big data.

4.5.2 Storage Cost Analysis

The storage cost refers as to the amount of storage space of the TT cores decomposed by TNs. Given an Nth-order tensor $\underline{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, suppose that $I = max\{I_n\}(n = 1, \cdots, N)$, and $r = max\{r_n\}(n = 0, 1, \cdots, N)$. According to the tensor network diagram in Fig. 2, it can be seen that the storage cost of a TT format is $\mathcal{O}(NIr^2)$. But the storage cost for the original tensor is $\mathcal{O}(I^N)$. It can be seen that the number of parameters of TT decomposition is asymptotically equal to that of CP decomposition [11]. Notably, if the tensor has low-rank structure or truncations are applied, the TT-ranks shall keep small, and the storage and computational complexities should be substantially reduced, which will play a significant role in storing and handling the cyber-physical-social big data.

Besides, QTT is adopted in some scenarios, especially for tensors with very large dimensionality, i.e., very large-scale vectors or matrices. Given an Nth-order tensor $\underline{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with $I_n = q^{J_n}$, if all orders are simultaneously quantized, then a quantized tensor $\underline{Y} \in \mathbb{R}^{q \times q \times \cdots \times q}$ can be yielded, its order number is $J_1 + J_2 + \cdots + J_N$ and dimensionality is q. Suppose that the TT decomposition is executed on the quantized tensor $\underline{Y}$, then the storage cost of QTT format is $\mathcal{O}(Nlog_q(I)qr^2)$, where $I = max\{I_n\}(n = 1, \cdots, N)$, and $r = max\{r_n\}(n = 0, 1, \cdots, J_1 + J_2 + \cdots + J_N)$. In practical computation, r is typically small, q commonly is equal to $2, 3$ or 4, thus the storage cost of QTT can be further reduced to a logarithmic (sub-linear) storage requirement $\mathcal{O}(Nlog_q(I))$ from the original exponential storage requirement $\mathcal{O}(I^N)$, which is referred as to so-called super-compression [17]. The super-compression is quiet important for reducing the storage and computation complexity for huge cyber-physical-social big data.

5 EXPERIMENTS

To verify the performance of the proposed ITTD approach, a series of experiments are conducted on cloud platform in accordance with various categories of datasets. Furthermore, we compare the ITTD with NTTD from multiple perspectives, including execution time, approximation error, and storage space.

5.1 Experimental Design

The experiments are implemented through Java Toolkit AKKA and Python Package Numpy. All experiments are executed on a cloud platform which configures an Intel's 16-core Xeon E5-2630 processor with 2.4GHZ and a 125G memory. The experiments are conducted from two scenarios, one is implemented on the random data and the other is on the real-world CPSS data. The random data are generated according to different parameters, e.g., order, scale, and density. Therein, the scale refers to the tensor size, the density is the ratio of the number of nonzero elements to the number of total elements of the tensor, and the prescribed accuracy is the setting value during TT decomposition. The real-world CPSS data are derived from the public traffic system in Guangzhou city of China.

To compare the performance of ITTD, NTTD is selected as the baseline. In reality, the CPSS data decomposed by TT decomposition have been already stored in clouds or networks in TT format. Therefore, when new streaming data have generated, the NTTD should execute the following steps to obtain the TT format of the updated tensor. Firstly, the TT format of the original tensor should be reconstructed. Secondly, we should add the incremental tensor to the original tensor and generate the updated tensor. Finally, we decompose the updated tensor according to TT decomposition and obtain the ultimate TT format.

5.2 Simulations and Evaluations

To verify the performance of the proposed ITTD approach and more fully reflect the influence of various kinds of data, we conduct a series of experiments by generating some random data according to different scales, densities, and prescribed accuracies. In these experiments, the tensor orders are set to 7th-order, 8th-order, and 9th-order. Their original tensor sizes are $10*10*10*10*10*10*2$ (denoted as 10^6*2), $10*10*10*10*10*10*10*2$ (denoted as 10^7*2), and $10*10*10*10*10*10*10*10*2$ (denoted as 10^8*2), respectively. And every incremental tensor sizes are 10^6*1, 10^7*1, and 10^8*1, which shall be appended to the original tensor along the last order, until the updated tensors increase to 10^6*10, 10^7*10, and 10^8*10 after 8 increments. Three densities are set to 1, $1e-4$, and $1e-5$, and three prescribed accuracies are set to 0, $1e-2$, and $1e-1$. Therefore, 9 random datasets are generated according to different orders (i.e., $order = 7, 8, 9$) and densities (i.e., $d = 1, 1e-4, 1e-5$). Based on these datasets, we conduct 27 groups of experiments according to different prescribed accuracies (i.e., $eps = 0, 1e-2, 1e-1$). Afterwards, the comparisons between ITTD and NTTD under diverse situations are illustrated from the execution time, approximation error, and storage space perspectives.

5.2.1 Comparison of Execution Time

The comparisons of execution time between ITTD and NTTD under different cases are depicted in Figs. 9∼12, where the X-axis and Y-axis represent the tensor size and execution time, respectively. Fig. 9 illustrates the influence of different data scales to the execution time. It can be seen from Figs. 9(a)(b)(c) that ITTD is inferior to NTTD in execution time when the original tensor size is 10^6*2. However, when the original tensor size increases to 10^7*2 or 10^8*2, the performance of ITTD in execution time outperforms that of NTTD. And the noticeable trend is that the larger the scale is, the greater superiority the ITTD achieves. The experimental results are exactly consistent with the analysis in Eq. 16. Note that only partial experimental results are displayed in Fig. 9 because of the space limitations, and the remainder also follows the above conclusion.

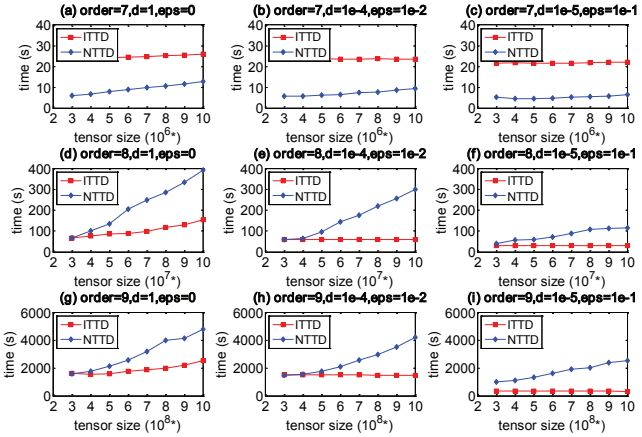


Fig. 9. The comparisons of execution time between ITTD and NTTD under different scales.

Fig. 10 demonstrates the influence of different prescribed accuracies to the execution time of ITTD and NTTD. In the 18 groups of experimental results, the general trend is that the larger the prescribed accuracy is, the less execution time the ITTD and NTTD cost. That is because that more noisy information are removed when the prescribed accuracy is larger. It can be seen from Figs. 9 and 10 that the execution time consumed by NTTD continuously grows up at a rapid rate with the increase of tensor size for all cases. However, if the prescribed accuracy is zero ($eps = 0$), i.e., there are no truncations in SVDs, the execution time spent by ITTD will also rise with the increase of tensor size, but the rise rate is much slower than NTTD. And if the prescribed accuracy is given ($eps = 1e-2$ or $1e-1$), i.e., the truncations are applied in SVDs, then the execution time of ITTD will remain nearly stable with the increase of tensor size. That is because that only the incremental tensor is decomposed and the orthogonalization and compression of low-order updated TT cores are executed in ITTD, and these operations shall take the same amount of time when the incremental tensors are the same size. The improved performance in execution time is very conductive to handling the cyber-physical-social big data.

Fig. 11 demonstrates the influence of different densities to the execution time of ITTD and NTTD. It shows a faint trend that the smaller the density is, the less time the ITTD

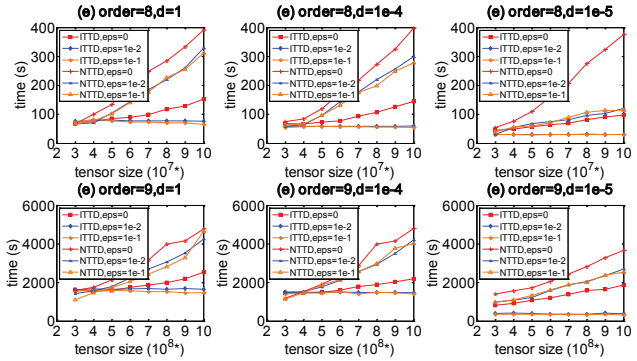


Fig. 10. The comparisons of execution time between ITTD and NTTD under different prescribed accuracies.

and NTTD spend, but the influence of different densities is not distinct. Actually, we can infer from Eq. 16 that the execution time of ITTD is related to tensor order N, dimensionality I, and TT rank r. Therefore, if the tensor size is given, the execution time will be determined by the TT rank. In general, the smaller the tensor's density, the smaller the TT rank of random tensor if the prescribed accuracy is given. But note that the TT rank of a tensor is determined by the tensor's own property when there is no truncation in SVDs (i.e., $eps = 0$), rather than the tensor's density.

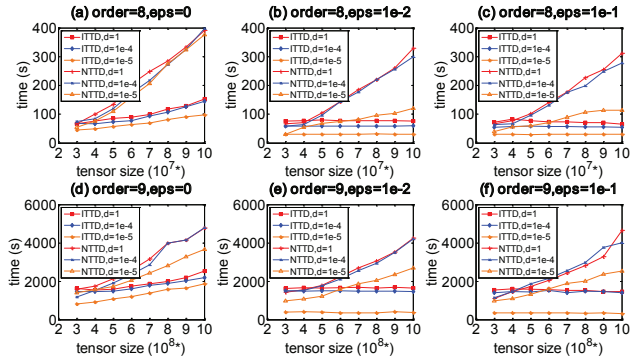


Fig. 11. The comparisons of execution time between ITTD and NTTD under different densities.

Fig. 12 depicts the influence of different orthogonalization and compression intervals to the execution time of ITTD. According to the explanation in section 4.4, the orthogonalization and compression procedure can be selectively executed in practical applications. In these experiments, four types of intervals (i.e., $INVL$=$1, 2, 4, 8$) are set, which means that the orthogonalization and compression procedure will be executed after every one, two, four, and eight incremental updates, respectively. Figs. 11(a)(b) and Figs. 11(c)(d) illustrate the independent and accumulated execution time of ITTD after every incremental update. It is obvious that ITTD will cost less execution time when the orthogonalization and compression procedure need not be executed. However, if the interval becomes larger, the execution time consumed by the last orthogonalization and compression will be very large, which may lead to larger accumulated execution time. It can be seen from Figs. 11(c)(d) that ITTD under the condition of $INVL$=1 costs the largest accumulated execution time, followed by $INVL$=$2, 4, 8$ (or $INVL$=$2, 8, 4$). Therefore, we can select an appropriate interval to execute the orthogonalization and compression procedure according to different practical situations.

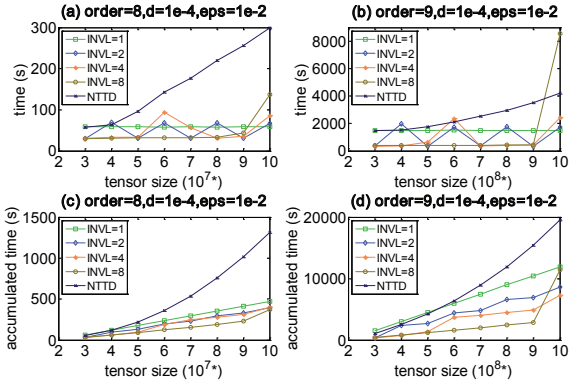


Fig. 12. The comparisons of execution time between ITTD and NTTD under different orthogonalization and compression intervals.

5.2.2 Comparison of Approximation Error

Fig. 13 illustrates the comparisons of the practical approximation error between ITTD and NTTD according to different prescribed accuracies. The approximation error is defined as $AE = \frac{\|\underline{X}-\widehat{\underline{X}}\|_F}{\|\underline{X}\|_F}$ [25], where $\underline{X}$ is the original tensor and $\widehat{\underline{X}}$ is the reconstructed tensor from the TT format. We select 12 groups of experiments based on two scales ($order = 8, 9$), two prescribed accuracies ($e = 1e-2, 1e-1$), and three densities ($d = 1, 1e-4, 1e-5$). It shows that the practical approximation errors of ITTD and NTTD are nearly equal and ITTD is slightly larger than NTTD. That is because ITTD executes two approximative operations in the TT decomposition for the incremental tensor and the compression for the ultimate TT cores, and NTTD executes the approximative operation only in the TT decomposition for the updated tensor. However, all approximation errors are less than the prescribed accuracies. When the prescribed accuracy is $1e-2$, their practical approximation errors in Figs. 13(b)(c)(f) and Figs. 13(a)(d)(e) is $1e-14$ and $1e-3$. When the prescribed accuracy is 0.1, their practical approximation errors are less than 0.1 and the average difference between ITTD and NTTD is only 0.005.

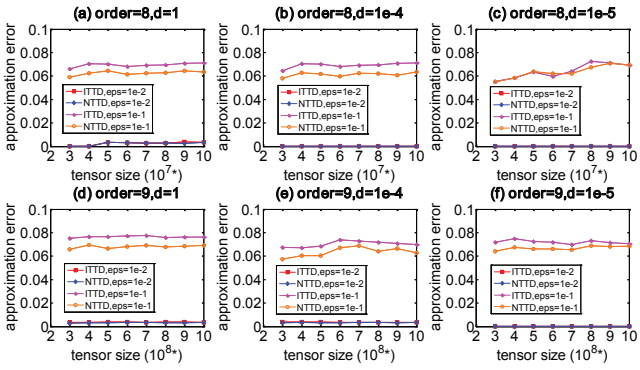


Fig. 13. The comparisons of approximation error between ITTD and NTTD under different situations.

5.2.3 Comparison of Storage Space

The comparisons of storage space between ITTD and NTTD under different situations are depicted in Fig. 14. It illustrates that ITTD and NTTD can save large amount of storage space compared with the original tensor, except for the first incremental update. The phenomenon are consistent with the storage analysis in section 4.5.2, namely, the advantage of TT decomposition in storage will become more prominent with the increment of the tensor order (N) and dimensionality (I). And if the prescribed accuracy is specified, we can find that the smaller the density is, the larger storage space the ITTD and NTTD save. That is because that the TT ranks become smaller when the data are sparser if the prescribed accuracy is specified. Besides, Fig. 14 shows that the differences of storage space between ITTD and NTTD are very subtle and nearly equal. Note that ITTD and NTTD take up less storage space with the increase of tensor size. That is because that ITTD once again compresses the original tensor during the compression for the updated TT cores, NTTD also compresses the reconstructed tensor again during the TT decomposition for the updated tensor.

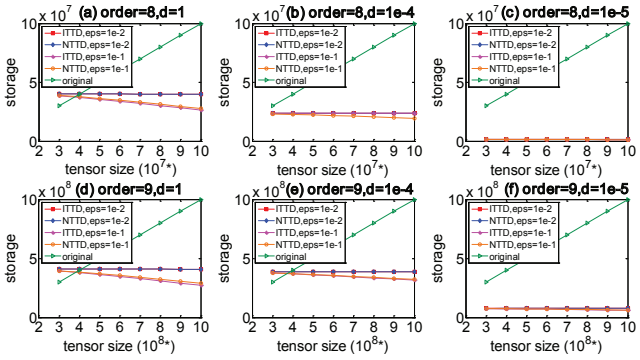


Fig. 14. The comparisons of storage space between ITTD and NTTD under different situations.

From the abundant experimental results, it is obvious that ITTD is significantly superior to NTTD in execution time on the premise of guaranteeing the nearly identical approximation error and storage space. The advantage is beneficial to improve the analysis efficiency for cyber-physical-social big data.

5.3 Case Study on CPSS Data

To further verify the performance of the proposed framework and ITTD approach on CPSS data, another group of experiments are conducted. The real-world CPSS data derive from the public traffic system in Guangzhou city of China. The dataset consists of 8 million bus card records generated by 2 million users and 4 bus lines from Aug. 1st, 2014 to Dec. 31st, 2014[1]. Firstly, we removed some incomplete bus card records. Then we integrate the remaining records (including bus lines, card types, traffics, time, etc.) and other data (including weather, temperature, etc.) to the fusion data. Afterwards, the characteristics of these data is analyzed, such as the value range, mean, variance, etc. Finally, the original CPSS tensor is constructed after preprocessing these CPSS data.

1. https://tianchi.aliyun.com/competition/information.htm?spm=5176.100067.5678.2.IsTw3H&raceId=231514

TABLE 1
The description of each order of bus traffic tensor

Order	Dimensionality	Description
Line	4	bus No. 6,10,11,15
Cardtype	3	student card, old card, ordinary card
Traffic	60	traffic is in $0 \sim 1200$, interval is 20
Weather	6	sunny, cloudy, light rain, moderate rain, heavy rain, snow
Temp	10	temperature is in $-10°C \sim 40°C$, interval is $5°C$
Hour	24	$1, 2, \cdots, 24$
Month	12	Jan., Feb., $\cdots$, Dec.
Day	7	Mon., Tues., $\cdots$, Sun.

Therefore, we construct an 8th-order tensor $\underline{X} \in \mathbb{R}^{4\times3\times60\times6\times10\times24\times12\times7}$ according to the fusion data. The corresponding information of each order of tensor $\underline{X}$ is explained in Table 1, and the entry of tensor $\underline{X}$ represents the swiped count. In the experiment, the streaming data tensor will be appended to the original tensor along the *day* order. To compare their performance of ITTD and NTTD, the data during the first week in August are selected as the test data. The initial tensor size is $4 \times 3 \times 60 \times 6 \times 10 \times 24 \times 12 \times 1$, and every incremental tensor size is $4 \times 3 \times 60 \times 6 \times 10 \times 24 \times 12 \times 1$, it will be gradually appended to the original tensor until the ultimate tensor size reaches $4 \times 3 \times 60 \times 6 \times 10 \times 24 \times 12 \times 7$ after 6 increments. Afterwards, we conduct a series of experiments according to four different prescribed accuracies (i.e., $eps = 0, 1e-5, 1e-2, 1e-1$) and compare the execution time, approximation error, and storage space of ITTD and NTTD. The comparisons are illustrated in Fig. 15, where the X-axis represents the *day* order of tensor and Y-axis represents the execution time, approximation error, and storage space, respectively.

It can be seen from Figs. 15(a)$\sim$(d) that ITTD outperforms NTTD in the execution time and the superiority becomes more prominent with the increase of tensor size. If the truncation is applied (e.g., Figs. 15(b)$\sim$(d)), the execution time of ITTD maintains a linear growth and even remains nearly constant with the increase of tensor size, however, NTTD keeps an exponential growth. Figs. 15(e)$\sim$(h) show the comparisons of approximation errors between ITTD and NTTD, we can see that the average approximation errors are much less than the prescribed accuracies and their differences are nearly negligible. For example, their average difference of approximation errors is $1e-14$ when the prescribed accuracy is $1e-5$. Figs. 15(i)$\sim$(l) demonstrate the comparisons of storage space between ITTD and NTTD. Note that the Y-axis represents the logarithm of the storage space. We can see that ITTD and NTTD maintain the same order of magnitude. As shown in Fig. 15(i), the storage space of ITTD and NTTD slightly exceed that of the original tensor when the prescribed accuracy is zero (i.e., $eps = 0$), that is because no truncation is applied during SVDs. However, the storage space of ITTD and NTTD are far less than that of the original tensor when the truncation is applied in SVDs (i.e., $eps = 1e-5, 1e-2, 1e-1$ in Figs. 15(j)$\sim$(l)). Experimental

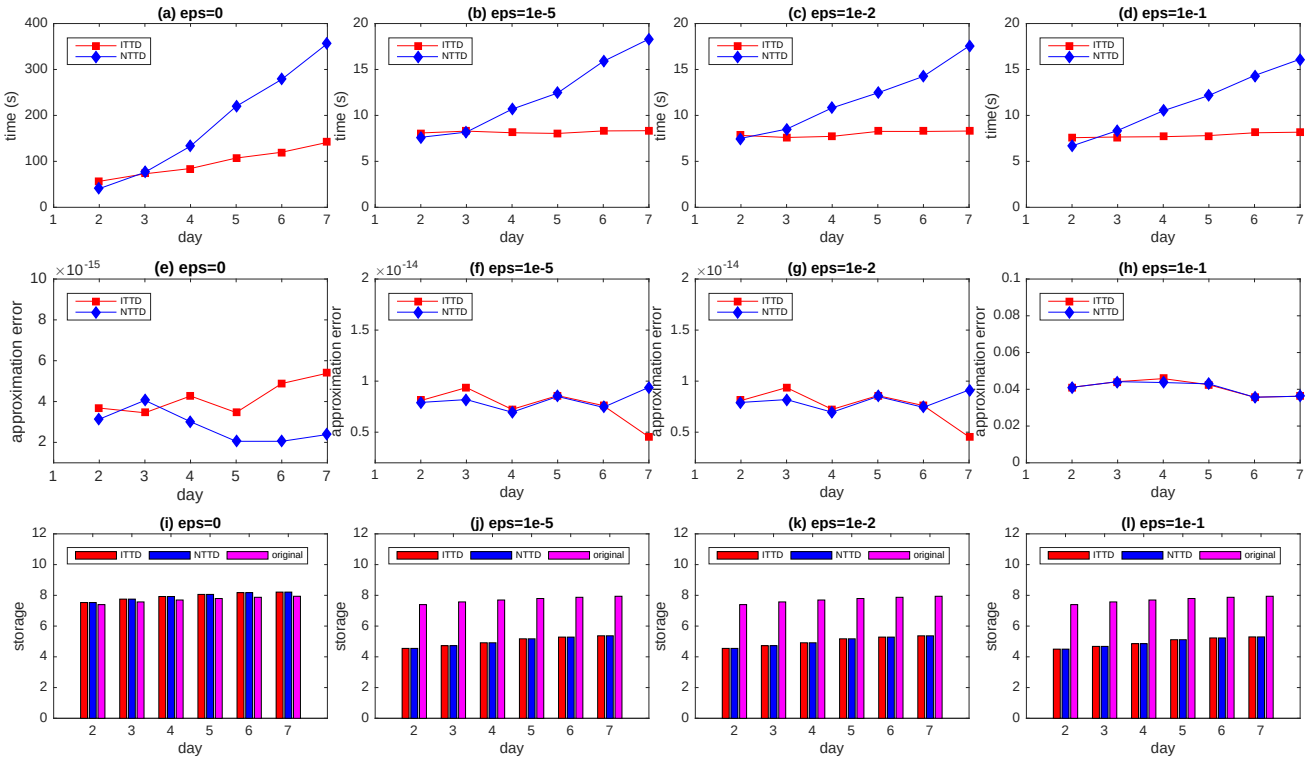


Fig. 15. The comparisons between ITTD and NTTD on CPSS data from execution time, approximation error, and storage space perspectives.

results demonstrate that the proposed ITTD approach also demonstrably outperforms the NTTD in reducing the execution time on real-world data.

6 Conclusions

In this paper, we present an effective processing framework based on tensor networks to represent, store, analyze, and handle cyber-physical-social big data. Tensor networks have some outstanding advantages in big data analysis, such as powerful data compression, distributed data storage, ubiquitous tensor computation, etc. TT decomposition, as the simplest yet efficient format of tensor networks, is illustrated in detail and applied to handle cyber-physical-social big data. Besides, given that the cyber-physical-social big data are generally generated in a streaming way, this paper also proposes an incremental tensor train decomposition approach to avoid the repeated computation for the history data. A wealth of experimental results demonstrate that the proposed ITTD approach outperforms NTTD in execution time on the premise of guaranteeing the nearly equal approximation error and storage space. The proposed processing framework and incremental decomposition approach are conducive to improving the cyber-physical-social big data analysis efficiency under ubiquitous computing environment.

In the future, we shall extend ITTD to meet the multi-order incremental scenarios by improving the implementation approach for achieving the TT format of the zero-padding tensor in multi-order incremental cases. Besides, we shall study the parallel computation of tensor operations in TT format without transforming TT cores to the original tensor by exploiting the mathematical characteristics of TT format and taking advantage of the distributed TT cores. Finally, we shall further study the TTcore-based data analysis methods and their practical applications.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Nos. 61867002 and 71704160) and Fundamental Research Funds for the Central Universities (No. 2018KFYXKJC046).

References

[1] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computing Surveys*, vol. 46, no. 4, pp. 55–84, 2014.

[2] J. Zeng, L. T. Yang, and J. Ma, "A system-level modeling and design for cyber-physical-social systems," *ACM Transactions on Embedded Computing Systems*, vol. 15, no. 2, pp. 35–61, 2016.

[3] J. Wan, D. Zhang, S. Zhao, L. T. Yang, and J. Lloret, "Context-aware vehicular cyber-physical systems with cloud support: architecture, challenges, and solutions," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 106–113, 2014.

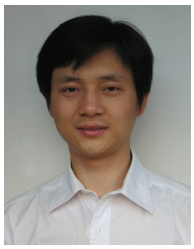
[4] D. Zhang, S. Zhao, L. T. Yang, M. Chen, Y. Wang, and H. Liu, "Nextme: localization using cellular traces in internet of things," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 2, pp. 302–312, 2015.

[5] S. Jiang, X. Qian, T. Mei, and Y. Fu, "Personalized travel sequence recommendation on multi-source big social media," *IEEE Transactions on Big Data*, vol. 2, no. 1, pp. 43–56, 2016.

[6] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 97–107, 2014.

[7] L. Kuang, L. T. Yang, and Y. Liao, "An integration framework on cloud for cyber physical social systems big data," *IEEE Transactions on Cloud Computing*, 2015.

[8] I. Babuška, F. Nobile, and R. Tempone, "A stochastic collocation method for elliptic partial differential equations with random input data," *SIAM Journal on Numerical Analysis*, vol. 45, no. 3, pp. 1005–1034, 2007.
[9] A. Cichocki, "Tensor decompositions: a new concept in brain data analysis?" *Journal of Control, Measurement, and System Integration*, vol. 47, no. 7, pp. 507–517, 2011.
[10] H. Kasai, W. Kellerer, and M. Kleinsteuber, "Network volume anomaly detection and identification in large-scale networks based on online time-structured traffic tensor tracking," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 636–650, 2016.
[11] I. V. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.
[12] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.
[13] J. Håstad, "Tensor rank is NP-complete," *Journal of Algorithms*, vol. 11, no. 4, pp. 644–654, 1990.
[14] I. V. Oseledets and E. E. Tyrtyshnikov, "Breaking the curse of dimensionality, or how to use svd in many dimensions," *Society for Industrial and Applied Mathematics*, vol. 31, no. 5, pp. 3744–3759, 2009.
[15] W. Hackbusch and S. Kühn, "A new scheme for the tensor representation," *Fourier Analysis and Applications*, vol. 15, no. 5, pp. 706–722, 2009.
[16] L. Grasedyck and W. Hackbusch, "An introduction to hierarchical (h-)rank and tt-rank of tensors with examples," *Computational Methods in Applied Mathematics*, vol. 11, no. 3, pp. 291–304, 2011.
[17] B. N. Khoromskij, "O(dlogN)-quantics Approximation of N-d Tensors in High-dimensional Numerical Modeling," *Constructive Approximation*, vol. 34, no. 2, pp. 257–280, 2011.
[18] A. Cichocki, "Era of big data processing: A new approach via tensor networks and tensor decompositions," *arXiv preprint arXiv:1403.2048*, 2014. [Online]. Available: http://arxiv.org/abs/1403.2048
[19] B. N. Khoromskij and I. V. Oseledets, "Qtt approximation of elliptic solution operators in higher dimensions," *Russian Journal of Numerical Analysis and Mathematical Modelling*, vol. 26, no. 3, pp. 303–322, 2011.
[20] S. V. Dolgov, B. N. Khoromskij, I. V. Oseledets, and D. V. Savostyanov, "Computation of extreme eigenvalues in higher dimensions using block tensor train format," *Computer Physics Communications*, vol. 185, no. 4, pp. 1207–1216, 2014.
[21] D. Kressner and A. Uschmajew, "On low-rank approximability of solutions to high-dimensional operator equations and eigenvalue problems," *Linear Algebra and its Applications*, vol. 493, pp. 556–572, 2016.
[22] J. R. Bunch and C. P. Nielsen, "Updating the singular value decomposition," *Numerische Mathematik*, vol. 31, no. 2, pp. 111–129, 1978.
[23] X. Zhou, J. He, G. Huang, and Y. Zhang, "SVD-based incremental approaches for recommender systems," *Journal of Computer and System Sciences*, vol. 81, no. 4, pp. 717–733, 2015.
[24] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos, "Incremental tensor analysis: Theory and applications," *ACM Transactions on Knowledge Discovery from Data*, vol. 2, no. 3, p. 11, 2008.
[25] L. Kuang, F. Hao, L. T. Yang, M. Lin, C. Luo, and G. Min, "A tensor-based approach for big data representation and dimensionality reduction," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 3, pp. 280–291, 2014.
[26] N. Lee and A. Cichocki, "Fundamental tensor operations for large-scale data analysis in tensor train formats," 2014. [Online]. Available: http://arxiv.org/abs/1405.7786
[27] F. Verstraete, V. Murg, and J. I. Cirac, "Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems," *Advances in Physics*, vol. 57, no. 2, pp. 143–224, 2008.
[28] R. Orús, "A practical introduction to tensor networks: Matrix product states and projected entangled pair states," *Annals of Physics*, vol. 349, pp. 117–158, 2014.
[29] A. Cichocki, "Tensor networks for big data analytics and large-scale optimization problems," *arXiv preprint arXiv:1407.3124*, 2014. [Online]. Available: http://arxiv.org/abs/1407.3124
[30] S. Handschuh, "Numerical methods in tensor networks," *PhD thesis, University of Leipzig*, 2015.



Huazhong Liu received the BS degree in Computer Science from the School of Computer Science and Technology, Jiangxi Normal University, Nanchang, China, in 2004, and the MS degree in Computer Science from the College of Mathematics and Computer Science, Hunan Normal University, Changsha, China, in 2009. Currently, he is pursuing the PhD degree at the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. His research interests include big data, cloud computing, Internet of Things, and scheduling optimization.

Laurence T. Yang received the BE degree in Computer Science and Technology from Tsinghua University, China, and the PhD degree in Computer Science from University of Victoria, Canada. He is a professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, China, and with the Department of Computer Science, St. Francis Xavier University, Canada. His research interests include parallel and distributed computing, embedded and ubiquitous/pervasive computing, and big data. His research has been supported by the National Sciences and Engineering Research Council, Canada (NSERC), and the Canada Foundation for Innovation (CFI).

Yimu Guo received the BS degree in Computer Science from the College of Information, LiaoNing University, Shenyang, China, in 2015, and the MS degree in Computer Science from the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, in 2017. His research interests include big data and parallel computing.

Xia Xie received the BE degree in Computer Science and the PhD degree in Computer System Architecture from Huazhong University of Science and Technology, Wuhan, China, in 2002 and 2006, respectively. She is an associate professor at Services Computing Technique and System Lab, Big Data Technology and System Lab, Cluster and Grid Computing Lab, and School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. Her research interests include Big Data Mining, Performance Evaluation.

Jianhua Ma received the MS degree from the National University of Defense Technology, Changsha, China, in 1985, and the PhD degree from Xidian University, Xian, China, in 1990. He is currently a Professor with the Faculty of Computer and Information Sciences, Hosei University, Tokyo, Japan. He has published over 200 papers and edited over 20 books/proceedings and over 20 journal special issues. His current research interests include multimedia, networks, ubiquitous computing, social computing, and cyber intelligence.