STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **12**, July 2024, pp 1061–1075.
Published online in International Academic Press (www.IAPress.org)

Randomized Density Matrix Renormalization Group Algorithm for Low-rank Tensor Train Decomposition

Huilin Jiang, Zhongming Chen*, Gaohang Yu

Department of Mathematics, School of Sciences, Hangzhou Dianzi University, Hangzhou 310018, China

Abstract Tensor train decomposition is a powerful tool for processing high-dimensional data. Density matrix renormalization group (DMRG) is an alternating scheme for low-rank tensor train decomposition of large tensors. However, it may suffer from the curse of dimensionality due to the large scale of subproblems. In this paper, we proposed a novel randomized proximal DMRG algorithm for low-rank tensor train decomposition by using TensorSketch to alleviate the curse of dimensionality. Numerical experiments on synthetic and real-world data also demonstrate the effectiveness and efficiency of the proposed algorithm.

Keywords tensor train decomposition, randomized algorithm, proximal regularization, TensorSketch, DMRG

AMS 2010 subject classifications 15A69, 68W20, 49M27

DOI: 10.19139/soic-2310-5070-2030

1. Introduction

Tensors are multidimensional arrays and generalization of matrices, which are widely used in many fields such as machine learning, signal processing [19]. As real-world data explodes in volume and complexity, tensor decomposition has become a powerful tool for large-scale data analysis. The idea of tensor decomposition is to represent a high-dimensional tensor as a combination of multiple low-rank tensors, thus realising data downscaling and capture the correlations, patterns and important features in the data effectively. The main tensor decompositions include CP decomposition [15], Tucker decomposition [32], tensor train (TT) decomposition [26], tensor ring decomposition [39] and so on. CP decomposition provides a useful way to factorize a tensor into the sum of rank-1 tensors. Unfortunately, it is not reliable due to the difficulty of determining the number of rank-1 components. Tucker decomposition is more stable than CP decomposition, but it suffers from the curse of dimensionality. On the other hand, TT decomposition is not affected by the curse of dimensionality and is more reliable. In this paper, we mainly focus on TT decomposition which is becoming increasingly popular due to its stability and efficiency.

The tensor train decomposition can decompose a large tensor into the product of a series of third-order tensors. Common methods for computing tensor TT decompositions include TT-SVD, TT-ALS. The former is based on the truncated singular value decomposition (SVD) of auxiliary unfolding matrices, while the latter mainly updates each core tensor alternatively by solving corresponding least squares problem. The density matrix renormalization group (DMRG) is another efficient algorithm for low-rank tensor train decomposition, which was originally proposed to find the ground state of Hamiltonians of many-body quantum systems [17]. It proceeds in the same fashion as TT-ALS, but with one important difference: the optimization is performed over two neighboring cores. However, both methods may suffer from the curse of dimensionality (the data size of a tensor increases exponentially with

*Correspondence to: Zhongming Chen (Email: zmchen@hdu.edu.cn). Department of Mathematics, School of Sciences, Hangzhou Dianzi University, Hangzhou 310018, China.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

the increase of the dimensionality of the tensor). As datasets grow larger and larger, there is an increasing need for methods to handle them. One possible solution to the challenge is the use of randomization, which has been proven to be effective in computing the low-rank approximations of large-scale matrices [14, 21, 31, 4, 24].

Among the existing tensor decomposition methods, different randomized techniques have been applied to accelerate the low-rank approximations of tensors [5, 25, 13, 7, 8, 1, 12]. For TT decomposition, Che et al. proposed an adaptive randomized algorithm for computing the tensor train approximations of tensors [9]. Huber et al. proposed randomized TT decomposition which is a robust alternative to the classical deterministic TT-SVD algorithm at low computational expenses [16]. To make full use of TT format, Shi et al. proposed parallelizable sketching algorithms that compute the low-rank TT decomposition from various tensor inputs [29]. Yu et al. presented a randomized algorithm for low-rank tensor train approximation of tensors based on randomized block Krylov subspace iteration [38]. It is worth mentioning that most of methods are based on the randomized SVD for matrices [14], where the random Gaussian matrices are used. For large-scale tensors, this kind of methods is bottlenecked by the operation called the *tensor-times-matrix-chains*. To alleviate the computation cost, many works [36, 2, 11] has led to the technique of TensorSketch which is ideally suited for sketching Kronecker products. In this way, the random matrix is very sparse and the accuracy could be also guaranteed with high probability. Recently, the technique of TensorSketch has been used for computing low-rank approximations of CP decomposition [34], Tucker decomposition [22, 20], tensor train decomposition [10] and tensor ring decomposition [23, 37]. The main idea of these randomized algorithms is using TensorSketch to sketch the subproblems of alternating least squares (ALS). Motivated by the work of [10], we apply TensorSketch and DMRG scheme to compute the low-rank TT decomposition based on the regularized alternating least squares. Our contributions and the notations used in this paper are listed in Subsections 1.1 and 1.2, respectively.

1.1. Contributions

In this paper, we propose a randomized proximal DMRG algorithm for low-rank TT decomposition by using TensorSketch. The fast computation and approximation accuracy are also established. In summary, this paper has the following contributions:

- Based on the DMRG, a novel randomized algorithm is proposed for low-rank TT decomposition by using TensorSketch.
- Numerical experiments on synthetic and real-world data also demonstrate the effectiveness and efficiency of the proposed algorithm.

1.2. Notation

Throughout this paper, scalars are denoted by lower case letters, e.g. x; vectors are denoted by bold lower case letters, e.g. $\mathbf{x}$; matrices are denoted by capital letters, e.g. X; tensors of order 3 or higher are denoted by calligraphic letters, e.g. $\mathcal{X}$. For any positive integer n, denote $[n] = \{1, 2, \ldots, n\}$. For any matrix $A \in \mathbb{R}^{m\times n}$, the ith row vector and jth column vector of A are denoted by $A(i,:)$ and $A(:,j)$, respectively. The Kronecker product of two matrices is denoted with "$\otimes$". The identity matrix of size $n \times n$ is denoted by I_n.

We use the multi-index $\overline{i_1 i_2 \cdots i_d}$ to denote the element in $\left[\prod_{k=1}^{d} n_k\right]$ such that

$$\overline{i_1 i_2 \cdots i_d} = i_1 + (i_2 - 1)n_1 + \cdots + (i_d - 1)n_1 \cdots n_{d-1}$$

for any $i_j \in [n_j]$ and $j \in [d]$. For any 3rd-order tensor $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$, the ith slice of $\mathcal{A}$ is denoted by $\mathcal{A}(i) \in \mathbb{R}^{n_1\times n_3}$ for $i \in [n_2]$, the left unfolding $\mathcal{A}^{\mathrm{L}} \in \mathbb{R}^{n_1 n_2\times n_3}$ is defined as $\mathcal{A}^{\mathrm{L}}(\overline{i_1 i_2}, i_3) = \mathcal{A}(i_1, i_2, i_3)$ and the right unfolding $\mathcal{A}^{\mathrm{R}} \in \mathbb{R}^{n_1\times n_2 n_3}$ is defined as $\mathcal{A}^{\mathrm{R}}(i_1, \overline{i_2 i_3}) = \mathcal{A}(i_1, i_2, i_3)$.

For any tensor $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$, the mode-$k$ matricization $\mathcal{A}_{(k)} \in \mathbb{R}^{n_k\times\prod_{i\neq k} n_i}$ is defined as $\mathcal{A}_{(k)}(i_k, j) = \mathcal{A}(i_1, i_2, \ldots, i_d)$, where

$$j = 1 + \sum_{\substack{s=1\\ s\neq k}}^{d} (i_s - 1) \prod_{\substack{t=1\\ t\neq s}}^{s-1} n_t,$$

and the mode-$(k,k+1)$ matricization $\mathcal{A}_{(k,k+1)} \in \mathbb{R}^{n_k n_{k+1} \times \prod_{i \neq k,k+1} n_i}$ is defined as $\mathcal{A}_{(k,k+1)}\left(\overline{i_k i_{k+1}}, \overline{i_1 \cdots i_{k-1} i_{k+2} \cdots i_d}\right) = \mathcal{A}(i_1, i_2, \ldots, i_d)$ for $i_j \in [n_j]$ and $j \in [d]$. The Frobenius norm of $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is defined as $\|\mathcal{A}\|_F = \sqrt{\sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} \mathcal{A}(i_1, i_2, \ldots, i_d)^2}$.

Definition 1 (k-mode product [19])
The k-mode product of a tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ and a matrix $A \in \mathbb{R}^{m \times n_k}$ is denoted by $\mathcal{X} \times_k A$ and is of size $n_1 \times \cdots \times n_{k-1} \times m \times n_{k+1} \times \cdots \times n_d$ with each element given by

$$(\mathcal{X} \times_k A)(i_1, \ldots, i_{k-1}, j, i_{k+1}, \ldots, i_d) = \sum_{i_k=1}^{n_k} \mathcal{X}(i_1, \ldots, i_{k-1}, i_k, i_{k+1}, \ldots, i_d) A(j, i_k).$$

Definition 2 (Face-splitting product [30])
Given $A \in \mathbb{R}^{m \times n_1}$ and $B \in \mathbb{R}^{m \times n_2}$, the face-splitting product $C = A \square B \in \mathbb{R}^{m \times n_1 n_2}$ is defined by the row-wise Kronecker product of matrices A and B, i.e.,

$$C(i,:) = A(i,:) \otimes B(i,:), \quad i \in [m].$$

Definition 3 (Slice-wise product[10])
Given $\mathcal{A} \in \mathbb{R}^{r_1 \times n \times r_2}$ and $\mathcal{B} \in \mathbb{R}^{r_2 \times n \times r_3}$, the slice-wise product $\mathcal{C} = \mathcal{A} \star \mathcal{B} \in \mathbb{R}^{r_1 \times n \times r_3}$ is defined by the slice-wise product of tensors $\mathcal{A}$ and $\mathcal{B}$, i.e.,

$$\mathcal{C}(i) = \mathcal{A}(i)\mathcal{B}(i), \quad i \in [n].$$

The above notations are summarized in Table 1.

Table 1. Description of notations.

Notation	Meaning
x	Scalar
$\mathbf{x}$	Vector
X	Matrix
$\mathcal{X}$	dth-order tensor $(d \geq 3)$
$[n]$	The set $\{1, 2, \ldots, n\}$
$A(i,:)$	The ith row vector of matrix A
$A(:,j)$	The jth column vector of matrix A
$\otimes$	Kronecker product
I_n	Identity matrix of size $n \times n$
$\mathcal{A}(i)$	The ith slice of 3rd-order tensor $\mathcal{A}$
$\mathcal{A}^{\mathrm{L}}$	The left unfolding of 3rd-order tensor $\mathcal{A}$
$\mathcal{A}^{\mathrm{R}}$	The right unfolding of 3rd-order tensor $\mathcal{A}$
$\mathcal{A}_{(k)}$	The mode-(k) matricization of tensor $\mathcal{A}$
$\mathcal{A}_{(k,k+1)}$	The mode-$(k,k+1)$ matricization of tensor $\mathcal{A}$
$\|\cdot\|_F$	Frobenius norm
$\times_k$	$k-$mode product
$\square$	Face-splitting product
$\star$	Slice-wise product

1.3. Organization

This paper is organized as follows. Some preliminaries are introduced in Section 2. In Section 3, we propose the randomized proximal DMRG algorithm for low-rank TT decomposition based on TensorSketch and analysis the

computational complexity and accuracy of our algorithm. In Section 4, numerical experiments for synthetic and real-world problems are presented to show the validity of proposed algorithm. Finally, the conclusions are drawn in Section 5.

2. Preliminaries

2.1. Tensor Train decomposition

The tensor train (TT) decomposition is to represent a high-order tensor as the product of a series of third-order tensors, which was originally proposed by Oseledets [26]. Specifically, the main idea of TT decomposition is to re-express each element of a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ as

$$\mathcal{A}(i_1, i_2, \cdots, i_d) = \mathcal{G}_1(i_1)\mathcal{G}_2(i_2)\cdots\mathcal{G}_d(i_d), \tag{2.1}$$

where $\mathcal{G}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}, k = 1, 2, \ldots, d$ are called TT-cores. Here we assume $r_0 = r_d = 1$ such that the product on the right-hand side of (2.1) is a scalar. The quantities $r_0, r_1, \ldots, r_d$ are called TT-ranks. The tensor $\mathcal{A}$ is also denoted by

$$\mathcal{A} = [\![\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_d]\!].$$

Let $n = \max\{n_1, n_2, \ldots, n_d\}$ and $r = \max\{r_0, r_1, \ldots, r_d\}$. It turns out that tensor train decomposition transforms the storage complexity of an n^d tensor into $O(dnr^2)$. The numerical stability of TT decomposition comes from the process of left and right orthogonalization [26]. Figure 1 illustrates the TT decomposition of a third-order tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$.

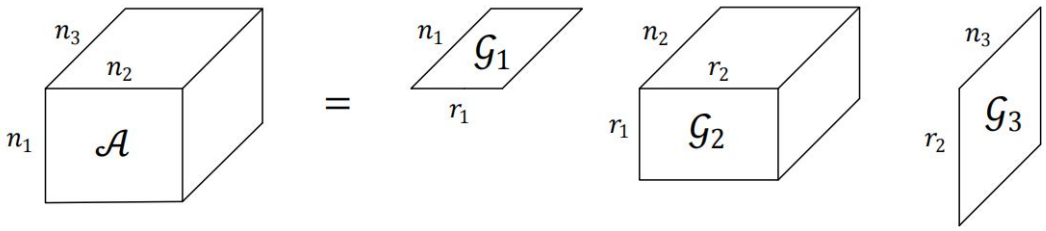

Figure 1. TT decomposition for $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$

2.2. DMRG scheme

The idea of the Density Matrix Renormalization Group (DMRG), proposed by White [35], was only later realized as an idea of constrained optimization in the matrix product. The DMRG method is an alternating least square method, but with one minor modification. Instead of minimizing over one core, the objective function is minimized over a pair of cores $(\mathcal{G}_k, \mathcal{G}_{k+1})$ simultaneously for $k \in [d-1]$. A new supercore $\mathcal{W}_k \in \mathbb{R}^{r_{k-1} \times n_k n_{k+1} \times r_{k+1}}$ is computed by

$$\mathcal{W}_k\left(\overline{i_k i_{k+1}}\right) = \mathcal{G}_k(i_k)\mathcal{G}_{k+1}(i_{k+1}) \tag{2.2}$$

for $i_k \in [n_k]$ and $i_{k+1} \in [n_{k+1}]$. This removes the information about the r_k. The optimization problem over $\mathcal{W}_k$ is a linear least squares problem. After $\mathcal{W}_k$ is updated, the supercore $\mathcal{W}_k$ is reshaped into the matrix $W_k \in \mathbb{R}^{r_{k-1} n_k \times n_{k+1} r_{k+1}}$. It follows that the TT-cores $\mathcal{G}_k$ and $\mathcal{G}_{k+1}$ are recovered from (2.2) by means of the singular value decomposition (SVD). The most important thing is that the rank r_k can be determined adaptively. This rank-reduction is called the decimation step of the DMRG scheme.

2.3. *TensorSketch*

TensorSketch is a technique for high-dimensional tensor approximation and dimensionality reduction. It restricts the hash map to a specific format, enabling fast multiplication of the sketching matrix with the chain of Kronecker products. The hash map in TensorSketch maps the indices of the tensor to a format that allows for efficient multiplication with the sketching matrix. This enables the algorithm to compute the sketch of a tensor quickly for various tasks, such as tensor decomposition and regression. Before introducing the definition of TensorSketch, we first give the definitions of CountSketch and k-wise independent hash map[18, 27].

Definition 4 (CountSketch)
The CountSketch matrix is definied as $S = \Omega D \in \mathbb{R}^{m\times n}$, where

(1) $h : [n] \to [m]$ is a hash map such that $Pr[h(i) = j] = \frac{1}{m}$ for all $i \in [n]$ and $j \in [m]$.
(2) $\Omega \in \mathbb{R}^{m\times n}$ is a matrix with $\Omega(j, i) = 1$ if $j = h(i)$ and $\Omega(j, i) = 0$ otherwise.
(3) $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix with diagonal a Rademacher vector $v \in \mathbb{R}^n$ (each entry is $+1$ or -1 with equal probability).

Definition 5 (k-wise independent)
A hash map $h : [n] \to [m]$ is called k-wise independent if the hash code of any fixed $i \in [n]$ is uniformly distributed in $[m]$, and the hash codes $h(i_1), h(i_2), \ldots, h(i_k)$ are independent random variables for any distinct $i_1, i_2, \ldots, i_k \in [n]$.

Note that there is a bijection between the set of indices $i \in \left[\prod_{k=1}^{d} n_k\right]$ and the d-tuples $(i_1, i_2, \ldots, i_d) \in [n_1] \times [n_2] \times \cdots \times [n_d]$ according to the reverse lexicographic order. For simplicity, we use the notation $f(i) = f(i_1, ..., i_d)$ for the funtion f on the domain $[n_1] \times [n_2] \times \cdots \times [n_d]$, where $i = 1 + \sum_{s=1}^{d}(i_s - 1)\prod_{t=1}^{s-1} n_t$.

Definition 6 (TensorSketch)
The order d TensorSketch matrix is defined as $S = \Omega D \in \mathbb{R}^{m\times\prod_{k=1}^{d} n_k}$, where

- $h : [n_1] \times [n_2] \times \cdots \times [n_d] \to [m]$ is the hash map

$$h(i_1, i_2, \ldots, i_d) = \left(\sum_{k=1}^{d}(h_k(i_k) - 1) \mod m\right) + 1, \tag{2.3}$$

  where $h_k : [n_k] \to [m]$ is a 3-wise independent hash map for $k = 1, 2, \ldots, d$.
- $\Omega \in \mathbb{R}^{m\times\prod_{k=1}^{d} n_k}$ is a matrix with $\Omega(j, i) = 1$ if $j = h(i)$ and $\Omega(j, i) = 0$ otherwise.
- $D \in \mathbb{R}^{\prod_{k=1}^{d} n_k\times\prod_{k=1}^{d} n_k}$ is a diagonal matrix with diagonal vector $v \in \mathbb{R}^{\prod_{k=1}^{d} n_k}$ given by

$$v(i_1, i_2, \ldots, i_d) = \prod_{k=1}^{d} v_k(i_k), \tag{2.4}$$

  where $v_k : [n_k] \to \{-1, 1\}$ is a 4-wise independent hash map for $k = 1, 2, \ldots, d$.

It is well-known that if h_k is 3-wise independent for $k \in [d]$, the hash map h constructed in TensorSketch is also 3-wise independent [6, 28].

3. Randomized DMRG algorithm for low-rank TT decomposition using TensorSketch

Given a tensor $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$ and TT-ranks $\{r_k\}_{k=0}^{d}$, the goal of low-rank tensor train decomposition is to minimize the objective function

$$f(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_d) = \frac{1}{2}\left\|[\![\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_d]\!] - \mathcal{A}\right\|_F^2 \tag{3.1}$$

where $\mathcal{G}_k \in \mathbb{R}^{r_{k-1}\times n_k\times r_k}$ for $k\in[d]$ and $r_0 = r_d = 1$. To minimize (3.1), DMRG algorithm updates each supercore $\mathcal{W}_k$ alternatively while the other TT-cores are fixed [35]. To be specific, for $k\in[d-1]$, the supercore is updated by solving the corresponding least squares problem, i.e.,

$$\min_{\mathcal{W}_k} \quad \frac{1}{2}\left\|\left(G_{>k+1}^\top \otimes G_{<k}\right)\mathcal{W}_{k(2)}^\top - \mathcal{A}_{(k,k+1)}^\top\right\|_F^2 \tag{3.2}$$

where $\mathcal{W}_k \in \mathbb{R}^{r_{k-1}\times n_k n_{k+1}\times r_{k+1}}$ are given by (2.2), $G_{<k} = \texttt{reshape}\left(\llbracket \mathcal{G}_1,\mathcal{G}_2,\ldots,\mathcal{G}_{k-1}\rrbracket, \prod_{i<k} n_i, r_{k-1}\right)$ and $G_{>k+1} = \texttt{reshape}\left(\llbracket \mathcal{G}_{k+2},\mathcal{G}_{k+3},\ldots,\mathcal{G}_d\rrbracket, r_{k+1}, \prod_{i>k+1} n_i\right)$. Here we define $G_{<1} = G_{>d} = 1$. Typically, the DMRG method converges quickly, and only a few sweeps (a sweep is the sequence of iterations where all pairs $(\mathcal{G}_k,\mathcal{G}_{k+1})$ were optimized) are required. However, the solution of (3.2) may be not unique. Motivated by the work of [10], proximal regularization is a powerful technique that enhances the training process by making the subproblems well-defined and promoting faster convergence. In this paper, we consider DMRG with proximal regularization:

$$\min_{\mathcal{W}_k} \quad \frac{1}{2}\left\|\left(G_{>k+1}^\top \otimes G_{<k}\right)\mathcal{W}_{k(2)}^\top - \mathcal{A}_{(k,k+1)}^\top\right\|_F^2 + \frac{\sigma}{2}\left\|\mathcal{W}_k - \mathcal{W}_k^{(t)}\right\|_F^2 \tag{3.3}$$

where $\mathcal{W}_k^{(t)}$ denotes $\mathcal{W}_k$ at the tth iteration. The cost of solving subproblem (3.3) is $O(n^d r^2)$ which is impractical for large-scale problems, where $n = \max\{n_1,n_2,\ldots,n_d\}$ and $r = \max\{r_0,r_1,\ldots,r_d\}$. So our idea is to find a sketching matrix $S\in\mathbb{R}^{m\times\prod_{i\neq k,k+1} n_i}$ to solve the sketched proximal least squares problem

$$\min_{\mathcal{W}_k} \quad \frac{1}{2}\left\|SH_{k,k+1}\mathcal{W}_{k(2)}^\top - S\mathcal{A}_{(k,k+1)}^\top\right\|_F^2 + \frac{\sigma}{2}\left\|\mathcal{W}_k - \mathcal{W}_k^{(t)}\right\|_F^2, \tag{3.4}$$

where $H_{k,k+1} = G_{>k+1}^\top \otimes G_{<k}$. It follows that the supercore $\mathcal{W}_k$ is updated by

$$\mathcal{W}_{k(2)}^{(t+1)} = \left(\mathcal{A}_{(k,k+1)}S^\top SH_{k,k+1} + \sigma\mathcal{W}_{k(2)}^{(t)}\right)\left(H_{k,k+1}^\top S^\top SH_{k,k+1} + \sigma I\right)^{-1}. \tag{3.5}$$

We can see that if $\sigma > 0$, (3.5) is always well-defined even though $H_{k,k+1}^\top S^\top SH_{k,k+1}$ is singular. Here we use TensorSketch to construct the sketching matrix S in (3.4). TensorSketch is a special type of CountSketch, where the hash map is restricted to a special format to allow fast multiplication of the sketching matrix with the chain of Kronecker products. Denote by $F\in\mathbb{R}^{m\times m}$ the Fourier transform matrix, i.e.,

$$F = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & w & \cdots & w^{m-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & w^{m-1} & \cdots & w^{2(m-1)} \end{bmatrix}$$

where $w = e^{-\mathbf{i}\cdot 2\pi/m}$. Similar with [10], we establish the fast computation of TensorSketch for DMRG algorithm. The only difference is that we consider $S(G_{>k+1}^\top \otimes G_{<k})$ instead of $S(G_{>k}^\top \otimes G_{<k})$.

Lemma 1 ([10])
Let $S\in\mathbb{R}^{m\times\prod_{i\in[d]\setminus\{k,k+1\}} n_i}$ be the TensorSketch matrix generated by CountSketch matrices $S_i\in\mathbb{R}^{m\times n_i}$, $i\in[d]\setminus\{k,k+1\}$. It holds that

$$S(G_{>k+1}^\top \otimes G_{<k}) = F^{-1}[(FS_{>k+1}G_{>k+1}^\top)\square(FS_{<k}G_{<k})],$$

where $S_{<k}\in\mathbb{R}^{m\times n_1\cdots n_{k-1}}$ is the TensorSketch matrix generated by CountSketch matrices $\{S_i\}_{i<k}$ and $S_{>k+1}\in\mathbb{R}^{m\times n_{k+2}\cdots n_d}$ is the TensorSketch matrix generated by CountSketch matrices $\{S_i\}_{i>k+1}$.

Lemma 2 ([10])
Let $S_{<k}$ and $S_{>k}$ be the TensorSketch matrices defined as Lemma 1. It holds that

$$FS_{<k}G_{<k} = [(\mathcal{G}_1 \times_2 FS_1)\star(\mathcal{G}_2\times_2 FS_2)\star\cdots\star(\mathcal{G}_{k-1}\times_2 FS_{k-1})]^{\mathrm{L}}$$

and

$$FS_{>k+1}G_{>k+1}^{\top} = \left[((\mathcal{G}_{k+2} \times_2 FS_{k+1}) \star (\mathcal{G}_{k+3} \times_2 FS_{k+2}) \star \cdots \star (\mathcal{G}_d \times_2 FS_d))^{\mathrm{R}}\right]^{\top}.$$

The two lemmas above show that the computations of $FS_{<k}G_{<k}$ and $FS_{>k+1}G_{>k+1}^{\top}$ could be divided into the slice-wise products of corresponding TT-cores, respectively. Next, we give the computational complexity of $S\left(G_{>k+1}^{\top} \otimes G_{<k}\right)$.

Theorem 1
Let $n = \max\{n_1, n_2, \ldots, n_d\}$ and $r = \max\{r_0, r_1, \ldots, r_d\}$. The products of $FS_{>k+1}G_{>k+1}^{\top}$ and $FS_{<k}G_{<k}$ could be computed at a cost of $O\left((m + m\log m + n)(d-2)r^2\right)$. As a result, the computation cost of $S\left(G_{>k+1}^{\top} \otimes G_{<k}\right)$ is $O\left((m + m\log m)(d-1)r^2 + n(d-2)r^2\right)$.

Proof. For $i \in [d] \setminus \{k, k+1\}$, the computation cost of $\mathcal{G}_i \times_2 S_i$ is $O(nr^2)$ due to the structure of CountSketch matrices. It follows that the computation cost of $\mathcal{G}_i \times_2 FS_i$ is $O(nr^2 + r^2 m\log m)$. By recursion, the slice-wise products of $(\mathcal{G}_1 \times_2 FS_1) \star \cdots \star (\mathcal{G}_{k-1} \times_2 FS_{k-1})$ and $(\mathcal{G}_{k+2} \times_2 FS_{k+2}) \star \cdots \star (\mathcal{G}_d \times_2 FS_d)$ could be computed at the cost of $O\left(m(k-1)r^2\right)$ and $O\left(m(d-k-1)r^2\right)$, respectively. By lemma 2, the total computation cost of $FS_{>k+1}G_{>k+1}^{\top}$ and $FS_{<k}G_{<k}$ is $O\left((m + m\log m + n)(d-2)r^2\right)$. Furthermore, the cost of face-splitting product $(FS_{>k+1}G_{>k+1}^{\top})\square(FS_{<k}G_{<k})$ is $O(mr^2)$. By lemma 1, the total computation cost of $S\left(G_{>k+1}^{\top} \otimes G_{<k}\right)$ is $O\left((m + m\log m)(d-1)r^2 + n(d-2)r^2\right)$ when adding the cost of inverse fast Fourier transform. □

In the following, we derive the theoretical result of sketch size for approximating the optimal value of (3.3) according to the result of [10].

Lemma 3 (TensorSketch for Regularized Least Squares [10])
Given a full-rank matrix $P \in \mathbb{R}^{n_1 n_2 \cdots n_q \times s}$ with $n_1 n_2 \cdots n_q > s$, $B \in \mathbb{R}^{n_1 n_2 \cdots n_q \times n}$, $C \in \mathbb{R}^{s \times n}$ and $\sigma > 0$, let $S \in \mathbb{R}^{m \times n_1 n_2 \cdots n_q}$ be the TensorSketch matrix defined as in Lemma 2. Denote $X_{opt} = \arg\min_X \frac{1}{2}\|PX - B\|_F^2 + \frac{\sigma}{2}\|X - C\|_F^2$ and $\tilde{X}_{opt} = \arg\min_X \frac{1}{2}\|SPX - SB\|_F^2 + \frac{\sigma}{2}\|X - C\|_F^2$. If $m = \max\left\{8s^2(2+3^q)/\delta,\ 8s(2+3^q)/(\epsilon\delta)\right\}$ where $\epsilon > 0$, $0 < \delta \leq 1$, the approximation

$$\frac{1}{2}\left\|P\tilde{X}_{opt} - B\right\|_F^2 + \frac{\sigma}{2}\left\|\tilde{X}_{opt} - C\right\|_F^2 \leq (1+\epsilon)\cdot OPT$$

holds with probability at least $1-\delta$, where $OPT = \frac{1}{2}\|PX_{opt} - B\|_F^2 + \frac{\sigma}{2}\|X_{opt} - C\|_F^2$.

Theorem 2
Let $r = \max\{r_0, r_1, \ldots, r_d\}$. If $m = \max\left\{8r^4(2+3^{d-2})/\delta,\ 8r^2(2+3^{d-2})/(\epsilon\delta)\right\}$, there is at least $1-\delta$ probability that $\mathcal{W}_k^{(t+1)}$ computed by (3.5) is a solution with a relative error of ϵ from the optimal value of (3.3). In particular, if $0 < \epsilon < 1/r^2$, the result holds if $m = 8r^2(2+3^{d-2})/(\epsilon\delta)$.

The proposed density matrix renormalization group (DMRG) algorithm for low-rank tensor train decomposition using TensorSketch is described in Algorithm 1. Based on the above analysis, we know that the computation cost of $S\left(G_{>k+1}^{\top} \otimes G_{<k}\right)$ goes linearly with the order d, whereas the naive matrix multiplication would cost $O(mn^{d-2}r^2)$ which goes exponentially with the order d. The special structure of TensorSketch matrices makes the computation more practical for large-scale problems. In fact, the mode products $\{\mathcal{G}_i \times_2 FS_i\}_{i\in[d]\setminus\{k,k+1\}}$ could be also computed in parallel to reduce the cost. Moreover, there is no need to store the whole tensor $\mathcal{A}$ since only a few fibers are used to compute $S\mathcal{A}_{(k,k+1)}^{\top}$ in (3.4). In particular, if $\mathcal{A}$ is sparse, the computation cost of $S\mathcal{A}_{(k,k+1)}^{\top}$ is $O\left(nnz(\mathcal{A})\right)$, where $nnz(\mathcal{A})$ denotes the number of nonzero elements of $\mathcal{A}$.

4. Experimental results

To show the efficiency of the proposed algorithm (denoted by DMRG-TS), we compare it with other two algorithms. The first one is the deterministic algorithm TT-DMRG [35], which serves as the baseline for low-rank tensor train decomposition. The second one is the randomized algorithm called DMRG-Random which is

Algorithm 1 DMRG algorithm for low-rank tensor train decomposition using TensorSketch

Input: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, TT-ranks $\{r_k\}_{k=1}^{d-1}$, tolerable error ϵ, sketch size m and $\sigma > 0$
Output: Approximate tensor $\widetilde{\mathcal{A}}$ with TT-cores $\{\mathcal{G}_k\}_{k=1}^{d}$

Initialize TT-cores $\{\mathcal{G}_k\}_{k=1}^{d}$ of prescribed ranks
while $err > \epsilon$ **do**
  Execute right-to-left orthogonalizion for $\{\mathcal{G}_k\}_{k=1}^{d}$
  Define TensorSketch operators $S_k \in \mathbb{R}^{m \times n_k}$, $k \in [d]$
  for $k = 1, 2, \ldots, d-1$ **do**
    Compute $S(G_{>k+1}^{\top} \otimes G_{<k})$ by $F^{-1}[(FS_{>k+1}G_{>k+1}^{\top}) \square (FS_{<k}G_{<k})]$
    Compute $S\mathcal{A}_{(k,k+1)}^{\top}$ by using some fibers of $\mathcal{A}$
    Update $\mathcal{W}_k$ according to (3.5)
    $W_k \leftarrow \texttt{reshape}(\mathcal{W}_k, r_{k-1} * n_k, n_{k+1} * r_{k+1})$
    $[U, S, V] \leftarrow$ compute SVD of W_k
    $\mathcal{G}_k \leftarrow \texttt{reshape}(U(:, 1:r_k), r_{k-1}, n_k, r_k)$
    $\mathcal{G}_{k+1} \leftarrow \texttt{reshape}(S(1:r_k, 1:r_k)V(:, 1:r_k)^{\top}, r_k, n_{k+1}, r_{k+1})$
  end for
  $\widetilde{\mathcal{A}} \leftarrow [\![\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_d]\!]$
  $err = \frac{\|\widetilde{\mathcal{A}} - \mathcal{A}\|_F^2}{\|\mathcal{A}\|_F^2}$
end while

motivated by [33], and the sketching matrix in (3.4) is chosen such that the rows of $H_{k,k+1}$ are chosen randomly with equal probability. In all three algorithms, the TT-cores are updated from left to right, and we developed our own implementations based on specific problems. All three algorithms take the same TT-ranks as input (the boundary ranks are set to 1). To ensure fairness, we use the third-order zero tensors as the initial core tensor for our experiments, with stopping criteria of either reaching the maximum number of iterations or the algorithm reaching a tolerable error. To avoid the singularity of subproblems, we add a regularization term to the subproblems. The accuracy evaluation for the algorithms is measured by computing the relative error (denoted by "err") between the original tensor and the approximate tensor, calculated by the following formula:

$$err = \frac{\left\|\widetilde{\mathcal{A}} - \mathcal{A}\right\|_F^2}{\|\mathcal{A}\|_F^2}.$$

All experiments were conducted by using Matlab R2016b on a computer with an AMD E2 7TH-GEN @2.20GHz CPU and 8 GB of RAM. We utilized the MATLAB Tensor Toolbox [3] to perform the experiments.

4.1. Experimental Results for Synthetic Data

In the first synthetic experiment, we randomly generate a fourth-order tensor $\mathcal{A}_1 \in \mathbb{R}^{50 \times 50 \times 50 \times 50}$ with TT-format, where the entries of each core are drawn independently from a standard normal distribution. For simplicity, the TT-ranks are equal, i.e., $r_1 = r_2 = r_3$. The true rank of the generated tensor is denoted by r_{true} while the target rank used in the algorithms is denoted by r. We set the sketch size $m = 100$, $r = r_{true} = 10$ and compare the numerical performance of DMRG-TS and DMRG-Random under the case $\sigma = 0$ and $\sigma = 0.5$ respectively. The numerical results are presented in Figure 2.

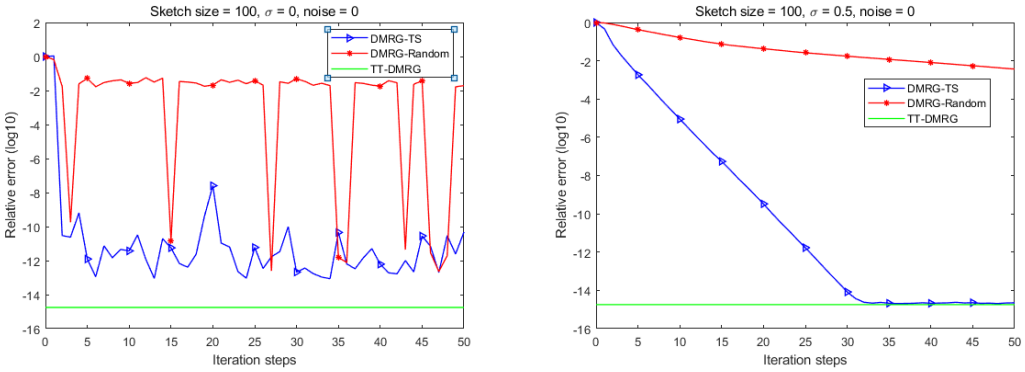

Figure 2. Iteration vs. relative error for randomly generated tensor $\mathcal{A}_1$ with target rank $r = r_{true} = 10$.

Figure 2 shows the relationship between the number of iterations and the relative error for DMRG-TS and DRMG-Random at the same sketch size. As shown in Figure 2, for both DMRG-TS and DRMG-Random, the relative error fluctuates sharply with the number of iteration steps when $\sigma = 0$ while the relative error converges steadily with the number of iteration steps when $\sigma = 0.5$. This indicates the advantage of proximal regularization.

Next, we randomly generate a sixth-order tensor $\mathcal{A}_2 \in \mathbb{R}^{10\times 10\times\cdots\times 10}$ with TT-format, where the entries of each core are drawn independently from a standard normal distribution. Besides, the generated tensor has been added by Gaussian noise with standard deviations of 0.1 and 0.01, respectively. The numerical results are reported in Figures 3 and 4. In addition, we also compared TT-TS which is proposed in [10] and DMRG-TS, and the corresponding numerical experimental results are shown in Figure 5.

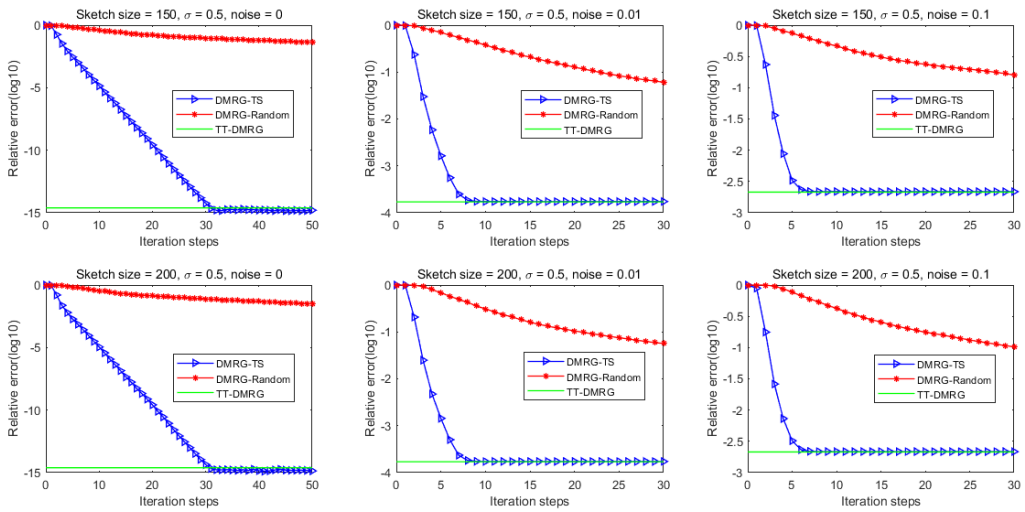

Figure 3. Iteration vs. relative error for randomly generated tensor $\mathcal{A}_2$ with target rank $r = r_{true} = 5$ and $\sigma = 0.5$.

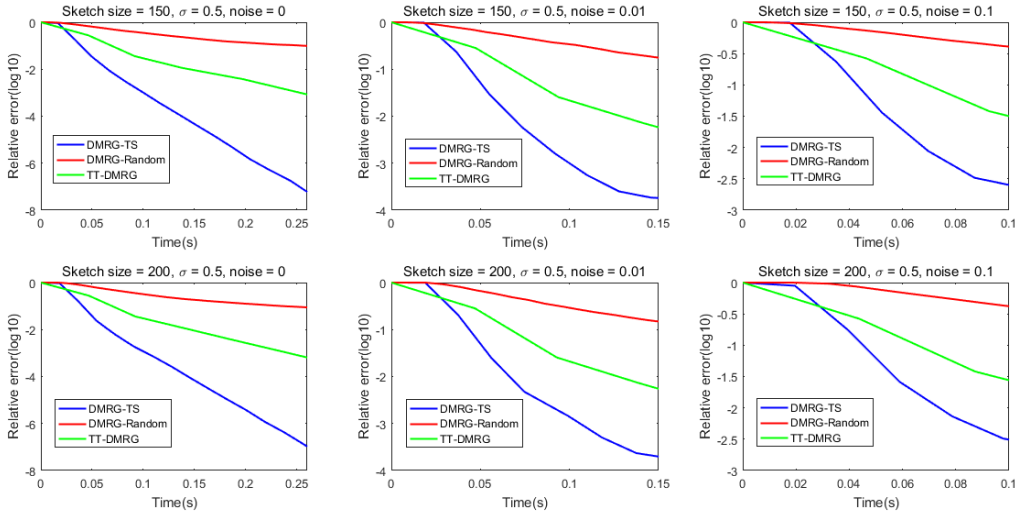

Figure 4. Time vs. relative error for $\mathcal{A}_2$ with target rank $r = r_{true} = 5$ and $\sigma = 0.5$.

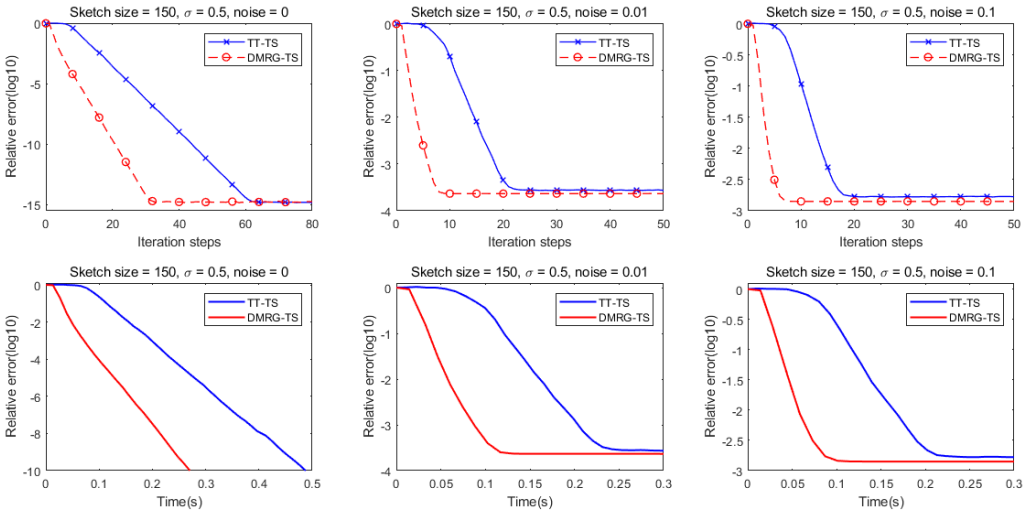

Figure 5. TT-TS vs. DMRG-TS for $\mathcal{A}_2$ with target rank $r = r_{true} = 5$ and $\sigma = 0.5$.

Figure 3 shows the relationship between the number of iterations and the relative error for DMRG-TS and DRMG-Random at the same sketch size, and all the data in the figures are the mean of 10 runs. As we can see, when the sketch size is low (sketch size = 150), our method (DMRG-TS) requires on average only 30 iterations to achieve an accuracy close to that of TT-DMRG, whereas DMRG-Random shows little improvement in terms of error reduction. When the sketch size is increased to 200, DMRG-Random also shows little improvement in terms of error reduction. Furthermore, in the presence of Gaussian noise, DMRG-Random requires more iterations than our method to achieve the same accuracy. In contrast, our method requires only a small number of samples to achieve an accuracy close to that of TT-DMRG. Figure 4 shows the relationship between the time and the relative error for DMRG-TS, DRMG-Random and TT-DMRG. From Figure 4, we can see that DMRG-TS requires

the least amount of time to achieve the approximation error when computing the TT decomposition of a large-scale tensor. This is because the complexity of our method is much lower than that of TT-DMRG, while its accuracy is much higher than that of DMRG-Random. Figure 5 shows that under the same conditions (sketch size, proximal parameter, Gaussian noise), both the number of iteration steps and time required for DMRG-TS to reach the convergence accuracy are less than that of TT-TS. This also demonstrates the superiority of DMRG-TS over TT-TS for low-rank TT decomposition.

4.2. Experimental Results for One-dimensional Functions

In the second experiment, we use DMRG-Random and DMRG-TS to approximate three one-dimensional functions $y = (x+1)\sin(100(x+1)^2)$, $y = x^{-\frac{1}{4}}\sin(\frac{2}{3}x^{\frac{3}{2}})$ and $y = \sin(\frac{4}{x})\cos(x^2)$, which are chosen from the highly oscillatory functions considered in [23]. We evaluated these three functions at 4^{10} points within the intervals $[-1, 1]$, $[0, 100]$ and $[0, 1]$, respectively. Then, we used the command `reshape` in MATLAB to transform the function values within the intervals into tenth-order tensors, denoted as $\mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5 \in \mathbb{R}^{4\times4\times\cdots\times4}$, respectively. During the approximation process, we set the target rank $r = 15$, $\sigma = 0.5$ and sketch size $m = 150$ uniformly. The numerical results are shown in Figure 6, which demonstrate the accuracy of the approximation using DMRG-TS and DMRG-Random after 25 iterations. Additionally, Figure 7 shows the relationship between the time and relative error for the three algorithms with $\sigma = 0.5$.

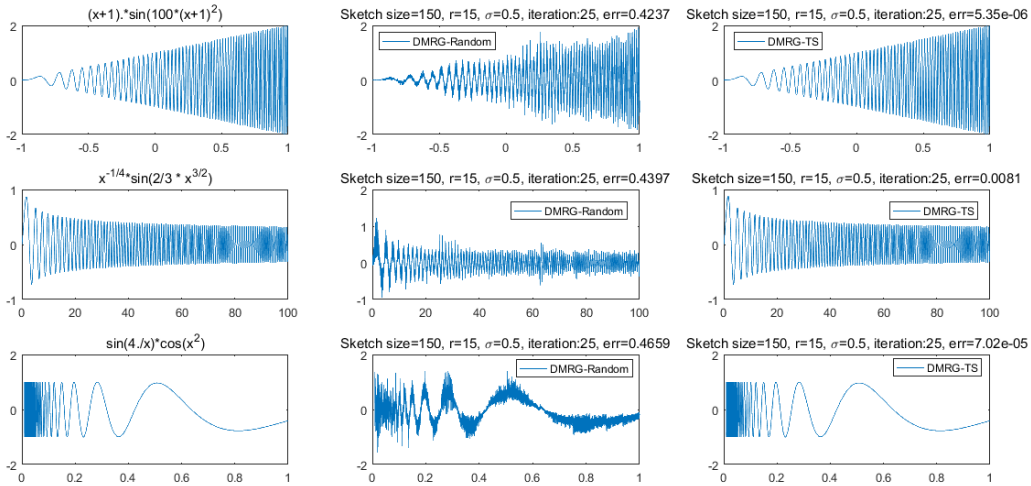

Figure 6. One-dimensional function approximation for $\mathcal{A}_3$, $\mathcal{A}_4$ and $\mathcal{A}_5$ with $\sigma = 0.5$.

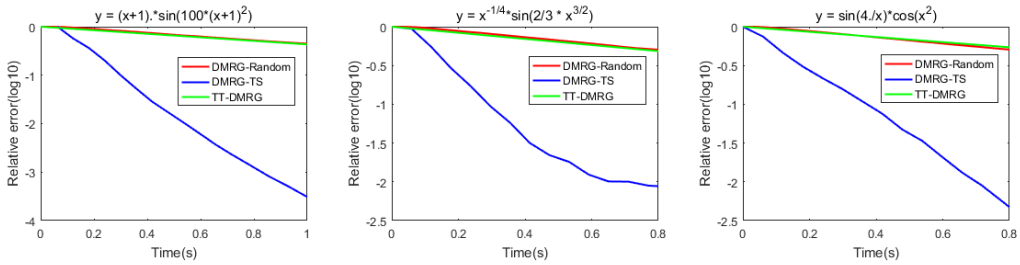

Figure 7. Time vs. relative error for $\mathcal{A}_3$, $\mathcal{A}_4$ and $\mathcal{A}_5$ with $\sigma = 0.5$.

The experimental results show that the approximations obtained from the DMRG-TS algorithm are more accurate than that obtained from the DMRG-Random algorithm at the same number of iterations. To sum up, under the same experimental conditions, our method outperforms the DMRG-Random algorithm. In terms of time cost, the DMRG-TS algorithm takes the least amount of time to achieve an accuracy comparable to that of the TT-DMRG algorithm. Thus we can conclude that our method is fast and efficient for the low-rank TT approximation of one-dimensional functions.

4.3. Experimental Results for Real Data

In this section, we consider three real datasets consisting of hyperspectral and video data. Next, we provide a brief overview of the data required for the experiments, which is summerized in Table 2.

- The first data is $Indian\_pines$, which is a hyperspectral image dataset of size $145 \times 145 \times 220$ sourced from Hyperspectral Remote Sensing Scenes[†]. It is a third-order tensor containing hyperspectral images, where the first two dimensions represent the height and width of the image, and the third dimension represents the number of spectral bands.
- The second data is $Skate$, which is a video dataset of size $720 \times 1280 \times 3 \times 420$ sourced from Pixabay[‡]. It is a fourth-order tensor representing color video of a man surfing on the sea. The various dimensions of this tensor represent different aspects of the video data, including its resolution, color space, and frame rate. In our experiment, we selected the information from the first 10 frames.
- The third data is $Bench$, which is also a video dataset of size $1080 \times 1920 \times 3 \times 364$ sourced from Pixabay[§]. It is a fourth-order tensor representing color video of a man resting on a park bench. In our experiment, we selected the information from the first 10 frames.

Table 2. Size and type of real data.

Data	Size	Type
$Indian\_pines$	$145 \times 145 \times 220$	Hyperspectral Image
$Skate$	$720 \times 1280 \times 3 \times 10$	Video
$Bench$	$1080 \times 1920 \times 3 \times 10$	Video

For hyperspectral image data, we use the command `reshape` in MATLAB to transform the data into $145 \times 145 \times 2 \times 110$ and set the TT-ranks as $(1, 20, 20, 20, 1)$ and the experimental results are shown in Figure 8. In Figure 8, we consider the effect of the three algorithms on the approximation of the original data when the proximal parameter $\sigma = 0.5$, and the relationship between the time and relative error for the three algorithms with $\sigma = 0.5$ are shown in Figure 9. In the experiments, we draw the spectral curve of the hyperspectral image for pixel at position $(1, 1)$. The accuracy of approximation is measured by the relative error (denoted by "err" in the figures) between the original spectral curve and the approximate spectral curve. As can be seen from Figures 8 and 9, the approximation of DMRG-TS is always better than that of DMRG-Random under the same settings (the values of sketch size, proximal parameter and iteration number). As the sketch size increases, both the approximations of DMRG-TS and DMRG-Random become more accurate and the gap between DMRG-TS and TT-DMRG gets smaller and smaller.

[†]https://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes
[‡]https://pixabay.com/videos/skate-sport-water-action-exercise-110734/
[§]https://pixabay.com/videos/bench-park-people-rest-pause-114/

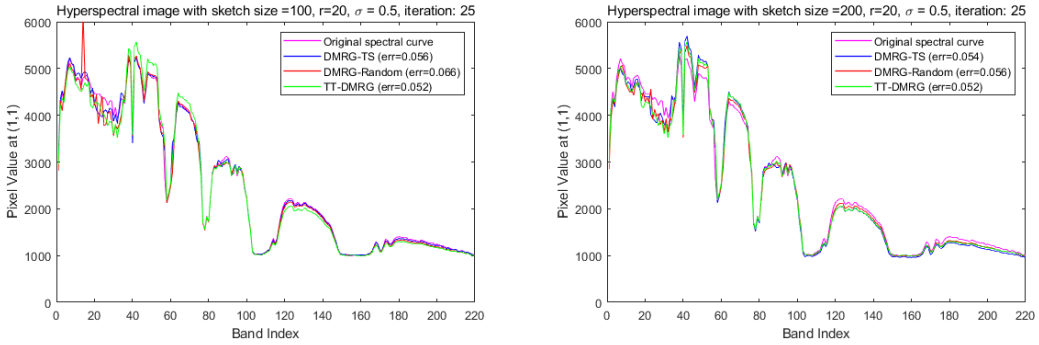

Figure 8. Numerical results for hyperspectral image with $\sigma = 0.5$.

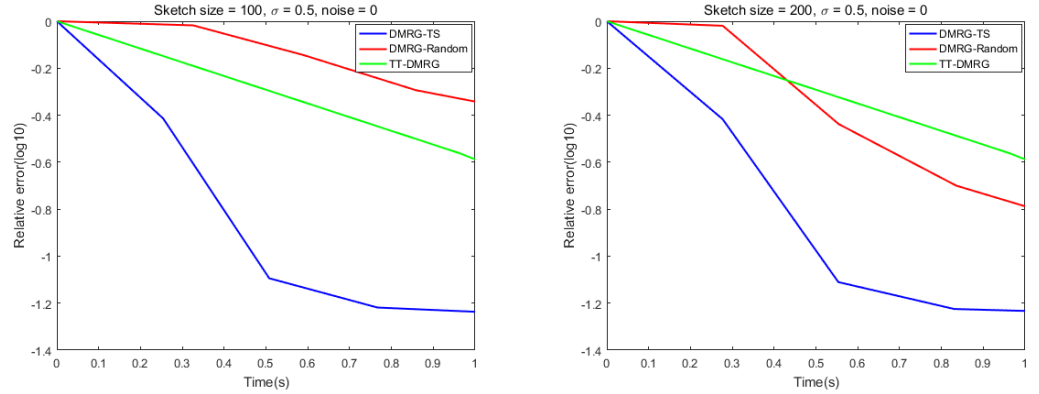

Figure 9. Time vs. relative error for hyperspectral image with $\sigma = 0.5$.

For the video data, we use the command `reshape` in MATLAB to transform the $Skate$ into $24 \times 30 \times 32 \times 40 \times 3 \times 10$ and $Bench$ into $27 \times 40 \times 48 \times 40 \times 3 \times 10$. Both of the datas are set the TT-ranks as $(1, 10, 10, 10, 10, 10, 1)$, and $\sigma = 0.5$. To verify the efficiency of randomized algorithms, we record the average computation time of each sweep for the tensors generated by the first 10 frames of the videos. The numerical results of the three algorithms are presented in Table 3. For visualization purpose, we only present the approximation results of the first frame of the video, as shown in Figure 10. From Table 3, we can see that DMRG-TS and DMRG-Random take much less time than TT-DMRG. According to Figure 10, the PSNR of DMRG-TS is greater than the DMRG-Random when $\sigma = 0.5$, which demonstrates the superiority of DMRG-TS over DMRG-Random under the same conditions.

Table 3. The average computation time of each sweep for video dataset with TT-ranks $r = (1, 10, 10, 10, 10, 10, 1)$ and $\sigma = 0.5$.

Data	$Skate$					$Bench$				
Data size	$24 \times 30 \times 32 \times 40 \times 3 \times 10$					$27 \times 40 \times 48 \times 40 \times 3 \times 10$				
Methods	TT-DMRG	DMRG-Random		DMRG-TS		TT-DMRG	DMRG-Random		DMRG-TS	
Sketch size	All	1000	2000	1000	2000	All	1000	2000	1000	2000
Time(s)	2.77	0.25	0.38	0.24	0.35	5.77	0.60	0.63	0.60	0.61

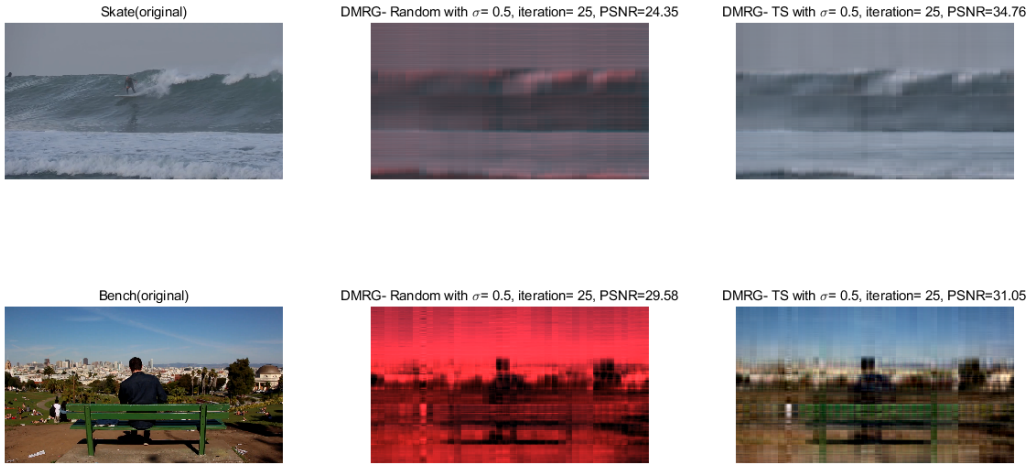

Figure 10. Numerical results for video experiment with sketch size $m = 1000$ and TT-ranks $r = (1, 10, 10, 10, 10, 10, 1)$.

5. Conclusion

In this paper, we proposed a novel randomized proximal DMRG algorithm for low-rank tensor train decomposition by using TensorSketch. Numerous experiments on both synthetic and real datasets were conducted to demonstrate the effectiveness and efficiency of the proposed algorithm. The numerical results showed the superiority of our algorithm in terms of computation complexity and accuracy for low-rank TT decomposition. On the other hand, TensorSketch is one of the randomized techniques to compute the low-rank approximations of large-scale tensors. Other efficient randomized techniques would be considered in the future.

Acknowledgements

This work was partially supported by National Natural Science Foundation of China (No. 12071104) and Natural Science Foundation of Zhejiang Province (No. LY22A010012).

REFERENCES

1. S. Ahmadi-Asl, S. Abukhovich, M. G. Asante-Mensah, A. Cichocki, A. H. Phan, T. Tanaka, and I. Oseledets, *Randomized algorithms for computation of Tucker decomposition and higher order SVD (HOSVD)*, IEEE Access, 9: 28684-28706, 2021.
2. H. Avron, H. Nguyen, and D. Woodruff, *Subspace embeddings for the polynomial kernel*, Advances in neural information processing systems, 27, 2014.
3. B. W. Bader, T. G. Kolda, and others, Tensor Toolbox for MATLAB, Version 3.5, www.tensortoolbox.org, February 25, 2023.
4. K. Batselier, W. Yu, L. Daniel, and N. Wong, *Computing low-rank approximations of large-scale matrices with the tensor network randomized SVD*, SIAM Journal on Matrix Analysis and Applications, 39(3): 1221-1244, 2018.
5. C. Battaglino, G. Ballard, and T. G. Kolda, *A practical randomized CP tensor decomposition*, SIAM Journal on Matrix Analysis and Applications, 39(2): 876-901, 2018.
6. J. L. Carter and M. N. Wegman, *Universal classes of hash functions*, Journal of Computer and System Sciences, 18(2): 143-154, 1979.
7. M. Che, Y. Wei, and H. Yan, *The computation of low multilinear rank approximations of tensors via power scheme and random projection*, SIAM Journal on Matrix Analysis and Applications, 41(2): 605-636, 2020.

8. M. Che, Y. Wei, and H. Yan, *Randomized algorithms for the low multilinear rank approximations of tensors*, Journal of Computational and Applied Mathematics, 390: 113380, 2021.
9. M. Che and Y. Wei, *Randomized algorithms for the approximations of Tucker and the tensor train decompositions*, Advances in Computational Mathematics, 45(1): 395-428, 2019.
10. Z. Chen, H. Jiang, G. Yu, and L. Qi, *Low-Rank Tensor Train decomposition Using TensorSketch*, arXiv preprint arXiv: 2309.08093, 2023.
11. H. Diao, Z. Song, W. Sun, and D. P. Woodruff, *Woodruff. Sketching for Kronecker product regression and p-splines*, In International Conference on Articial Intelligence and Statistics, pages 1299-1308. PMLR, 2018.
12. W. Dong, G. Yu, L. Qi, and X. Cai, *Practical sketching algorithms for low-rank Tucker approximation of large tensors*, Journal of Scientific Computing, 95(2): 52, 2023.
13. N. B. Erichson, K. Manohar, S. L. Brunton, and J. N. Kutz, *Randomized CP tensor decomposition*, Machine Learning: Science and Technology, 1(2): 025012, 2020.
14. N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM review, 53(2): 217-288, 2011.
15. F. L. Hitchcock, *The expression of a tensor or a polyadic as a sum of products*, Journal of Mathematics and Physics, 6(1-4): 164-189, 1927.
16. B. Huber, R. Schneider, and S. Wolf, *A randomized tensor train singular value decomposition*, In Compressed Sensing and its Applications: Second International MATHEON Conference 2015, pages 261-290. Springer International Publishing, 2017.
17. E. Jeckelmann, *Dynamical density-matrix renormalization-group method*, Physical Review B, 66(4), 045114, 2002.
18. D. M. Kane and J. Nelson, *Sparser Johnson-Lindenstrauss transforms*, Journal of the ACM (JACM), 61(1): 4:1–4:23, 2014.
19. T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM review, 51(3): 455-500, 2009.
20. L. Ma and E. Solomonik, *Fast and accurate randomized algorithms for low-rank tensor decompositions*, Advances in Neural Information Processing Systems, 34: 24299-24312, 2021.
21. M. W. Mahoney, *Randomized algorithms for matrices and data*, Foundations and Trends® in Machine Learning, 3(2): 123-224, 2011.
22. O. A. Malik and S. Becker, *Low-rank Tucker decomposition of large tensors using TensorSketch*, Advances in neural information processing systems, 31: 10096-10106, 2018.
23. O. A. Malik and S. Becker, *A sampling-based method for tensor ring decomposition*, In International Conference on Machine Learning, pages 7400-7411. PMLR, 2021.
24. P. G. Martinsson and J. A. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numerica, 29: 403-572, 2020.
25. R. Minster, A. K. Saibaba, and M. E. Kilmer, *Randomized algorithms for low-rank tensor decompositions in the Tucker format*, SIAM Journal on Mathematics of Data Science, 2(1): 189-215, 2020.
26. I. Oseledets, *Tensor train Decomposition*, SIAM Journal on Scientific Computing 33(5): 2295-2317, 2011.
27. R. Pagh, *Compressed matrix multiplication*, ACM Transactions on Computation Theory (TOCT), 5(3): 1–17, 2013.
28. M. Patrascu and M. Thorup, *The power of simple tabulation hashing*, Journal of the ACM (JACM), 59(3): 1-50, 2012.
29. T. Shi, M. Ruth, and A. Townsend, *Parallel algorithms for computing the tensor-train decomposition*, SIAM Journal on Scientific Computing, 45(3): C101-C130, 2023.
30. V. I. Slyusar. *Analytical model of the digital antenna array on a basis of face-splitting matrixs product* In Proceedings of International Conference on Antenna Theory and Techniques, pages 108-109, 1997.
31. J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Practical sketching algorithms for low-rank matrix approximation*, SIAM Journal on Matrix Analysis and Applications, 38(4): 1454-1485, 2017.
32. L. R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika 31(3): 279-311, 1966.
33. Y. Wu, R. Chen, and Z. Chen, *Solving Sylvester tensor equation based on tensor train decomposition (in Chinese)*, Journal of Hangzhou Dianzi University (Natural Sciences), 41(6): 94-99, 2021.
34. Y. Wang, H. Y. Tung, A. Smola, A. Anandkumar, *Fast and guaranteed tensor decomposition via sketching*, Advances in neural information processing systems, 28, 2015.
35. S. White, *Density-matrix algorithms for quantum renormalization groups*, Physical Review B, Vol. 48, no. 14, pp. 10345-10356, 1993.
36. D. P. Woodruff, *Sketching as a tool for numerical linear algebra*, arXiv preprint arXiv: 1411.4357, 2014.
37. Y. Yu and H. Li, *Practical sketching-based randomized tensor ring decomposition*, arXiv preprint arXiv: 2209.05647, 2022.
38. G. Yu, J. Feng, Z. Chen, X. Cai, and L. Qi, *A randomized block Krylov method for tensor train approximation*, arXiv preprint arXiv: 2308.01480, 2023.
39. Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki, *Tensor ring decomposition*, arXiv preprint arXiv: 1606.05535, 2016.