COMPUTING LOW-RANK APPROXIMATIONS OF LARGE-SCALE MATRICES WITH THE TENSOR NETWORK RANDOMIZED SVD

By

KIM BATSELIER, WENJIAN YU , LUCA DANIEL, AND NGAI WONG

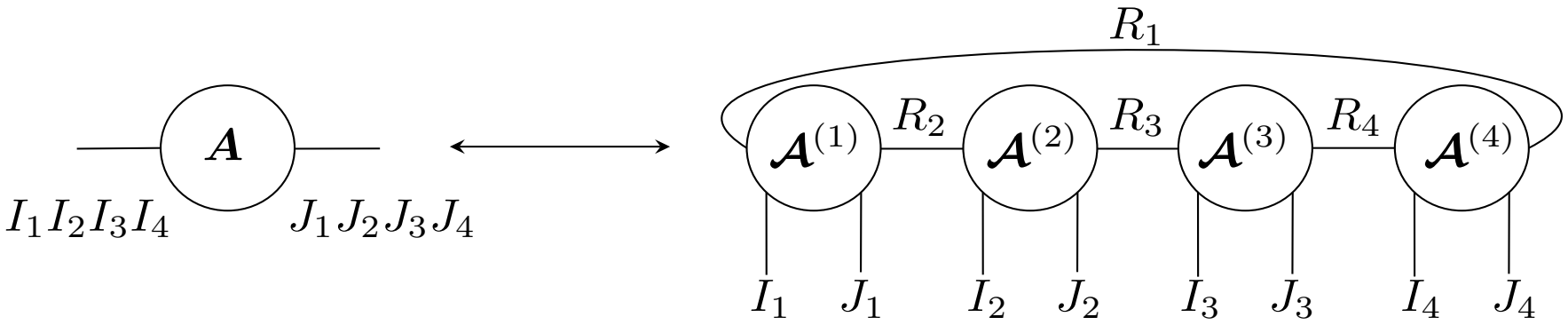
Beheshteh T. Rakhshan

Maziar Sargordi

Table of Contents

Preliminaries

- Matrix Product Operator (MPO)
  - Particular tensor network representation for matrices.
  - The matrix $\boldsymbol{A}$ of size $I_1\ I_2\ I_3\ I_4 \times J_1\ J_2\ J_3\ J_4$ can be represented as



  - If the MPO-ranks are all one then the matrix A is the outer product of core tensors.

Preliminaries

- **Theorem**

  A matrix $\boldsymbol{A}$ of size $I_1\ I_2\ I_3\ I_4 \times J_1\ J_2\ J_3\ J_4$ that satisfies

$$\boldsymbol{A} = \boldsymbol{A}^{(d)} \otimes \cdots \otimes \boldsymbol{A}^{(2)} \otimes \boldsymbol{A}^{(1)}$$

  has an MPO representation where the kth MPO tensor is $\boldsymbol{A}^{(k)}$ of size $1 \times I_k \times J_k \times 1, (k = 1, \dots, d)$ with unit canonical MPO ranks.

Converting a sparse matrix into an MPO

- The standard way to convert a matrix into MPO form is the TT-SVD algorithm
  - **Issues:**
    Computing the SVD of a sparse matrix destroys the sparsity
    Real-life matrices are typically so large that it is infeasible to compute their SVD

- We explain new conversion to MPO algorithm by following example
  Suppose we have a sparse matrix $A \in \mathbb{R}^{I \times J}$

$$\begin{pmatrix} 0 & 0 & 0 & \boldsymbol{A}_{14} \\ 0 & \boldsymbol{A}_{22} & 0 & 0 \\ 0 & 0 & \boldsymbol{A}_{33} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Converting a sparse matrix into an MPO

- The main idea of the method is to convert each nonzero block matrix into a rank-1 MPO and add them all together.

$$\begin{pmatrix} 0 & 0 & 0 & \boldsymbol{A}_{14} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \boldsymbol{E}_{14} \otimes \boldsymbol{A}_{14}$$

- Where $\boldsymbol{E}_{14} \in \mathbb{R}^{I_2 \times J_2}$ is a matrix of zeros except for $\boldsymbol{E}_{14}(1,4) = 1$.

$$\boldsymbol{A} = \boldsymbol{E}_{14} \otimes \boldsymbol{A}_{14} + \boldsymbol{E}_{22} \otimes \boldsymbol{A}_{22} + \boldsymbol{E}_{33} \otimes \boldsymbol{A}_{33}$$

Converting a sparse matrix into an MPO

- We can also partition $\boldsymbol{A}_{14}$ further into:

$$\boldsymbol{A}_{14} = \begin{pmatrix} 0 & 0 \\ \boldsymbol{X}_{14} & 0 \end{pmatrix} = \boldsymbol{E}_{21} \otimes \boldsymbol{X}_{14}$$

- Then the first term of
  $\boldsymbol{A} = \boldsymbol{E}_{14} \otimes \boldsymbol{A}_{14} + \boldsymbol{E}_{22} \otimes \boldsymbol{A}_{22} + \boldsymbol{E}_{33} \otimes \boldsymbol{A}_{33}$
  becomes $\boldsymbol{E}_{14} \otimes \boldsymbol{E}_{21} \otimes \boldsymbol{X}_{14}$ and likewise for the other terms:

$$\boldsymbol{A} = \boldsymbol{A}^{(d)} \otimes \cdots \otimes \boldsymbol{A}^{(2)} \otimes \boldsymbol{A}^{(1)}$$

Converting a sparse matrix into an MPO

ALGORITHM 4.1. *Sparse matrix to MPO conversion*
Input: *matrix* $\boldsymbol{A}$, *dimensions* $I_1, \ldots, I_d, J_1, \ldots, J_d$.
Output: *MPO* $\boldsymbol{\mathcal{A}}$ *with tensors* $\boldsymbol{\mathcal{A}}^{(1)}, \ldots, \boldsymbol{\mathcal{A}}^{(d)}$.

Initialize MPO $\boldsymbol{\mathcal{A}}$ *with zero tensors.*
for *all nonzero matrix blocks* $\boldsymbol{X} \in \mathbb{R}^{I_1 \times J_1}$ ***do***
Determine $d-1$ $\boldsymbol{E}_{ij}$ *matrices.*
Construct rank-1 MPO $\boldsymbol{\mathcal{T}}$ *with* $\boldsymbol{X}$ *and* $\boldsymbol{E}_{ij}$ *matrices.*
$\boldsymbol{\mathcal{A}} \leftarrow \boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{T}}$
end for

Converting a sparse matrix into an MPO

- Some properties of algorithm 4.1
  - How to partition the matrix $\boldsymbol{A}$ is optional.
  - The maximal number of core tensors in an MPO representation of $\boldsymbol{A}$ is max $(d_I, d_J) + 1$
  - The MPO obtained from Algorithm 4.1 has a uniform MPO-rank equal to the total number of nonzero matrix blocks $\boldsymbol{X}$ as determined by the partitioning of $\boldsymbol{A}$.
  - Since the addition of MPOs can be done by concatenation of the respective tensors, no actual computation is required, which allows a fast execution of Algorithm 4.1.

Tensor network randomized SVD

- The goal is to find a rank K factorization that consist of right and left singular values in MPO form and a diagonal matrix of singular values.

ALGORITHM 5.1. *Prototypical rSVD algorithm* [16, p. 227]

Input: *matrix* $\boldsymbol{A} \in \mathbb{R}^{I \times J}$, *target number* K, *oversampling parameter* s, *and exponent* q.

Output: *approximate rank-*$(K+s)$ *factorization* $\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$, *with* $\boldsymbol{U}, \boldsymbol{V}$ *orthogonal and* $\boldsymbol{S}$ *is diagonal and nonnegative.*

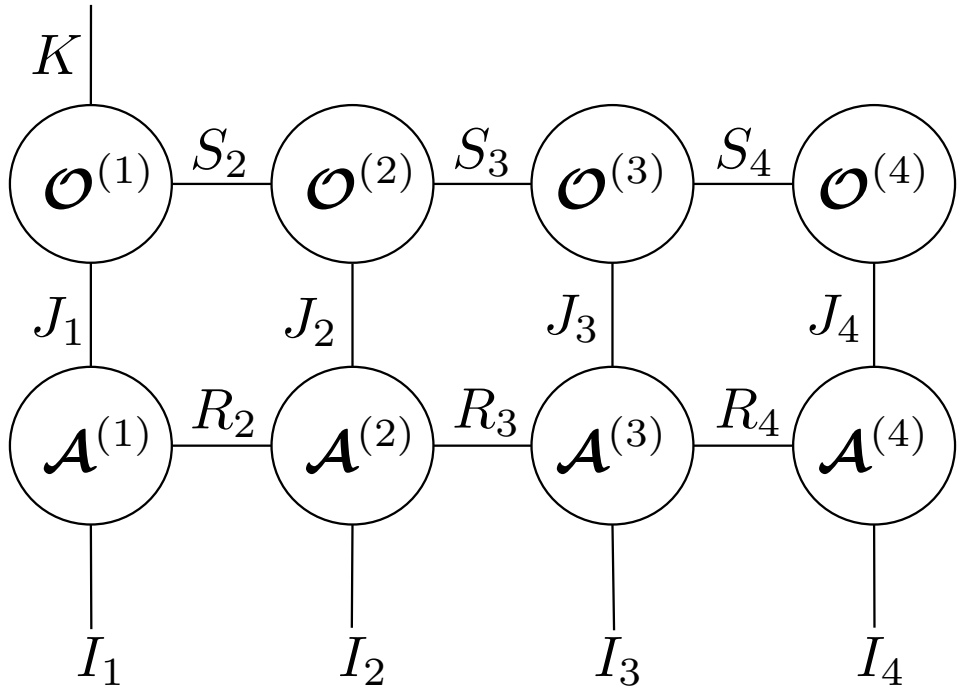
Generate an $J \times (K+s)$ *random matrix* $\boldsymbol{O}$.
$\boldsymbol{Y} \leftarrow (\boldsymbol{A}\boldsymbol{A}^T)^q \, \boldsymbol{A}\,\boldsymbol{O}$
$\boldsymbol{Q} \leftarrow$ *Orthogonal basis for the range of* $\boldsymbol{Y}$
$\boldsymbol{B} \leftarrow \boldsymbol{Q}^T \, \boldsymbol{A}$
Compute the SVD $\boldsymbol{B} = \boldsymbol{W}\boldsymbol{S}\boldsymbol{V}^T$.
$\boldsymbol{U} \leftarrow \boldsymbol{Q}\,\boldsymbol{W}$

Tensor network randomized SVD

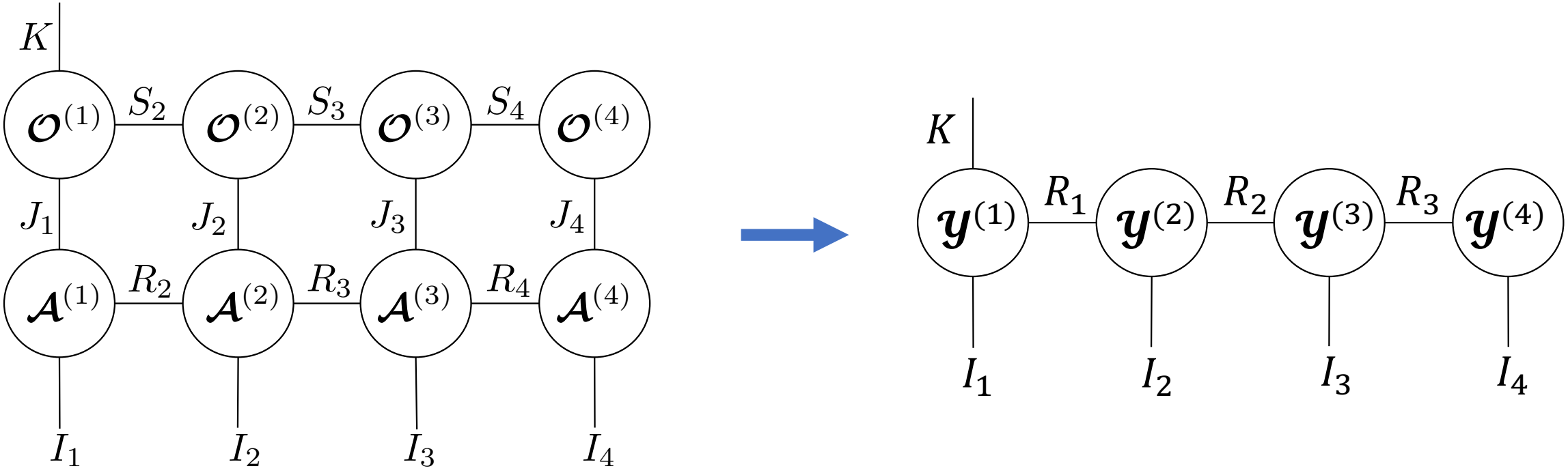
- **Lemma**
  A particular random $J_1 J_2 \ \ldots J_d \times K$ matrix $\boldsymbol{O}$ with rank($\boldsymbol{O}$) = K and $K \leq J_1$ can be represented by a unit-rank MPO with the following random MPO- tensors:

$$\mathcal{O}^{(1)} \in \mathbb{R}^{1 \times J_1 \times K \times 1},$$

$$\mathcal{O}^{(i)} \in \mathbb{R}^{1 \times J_i \times 1 \times 1} \qquad (2 \leq i \leq d).$$

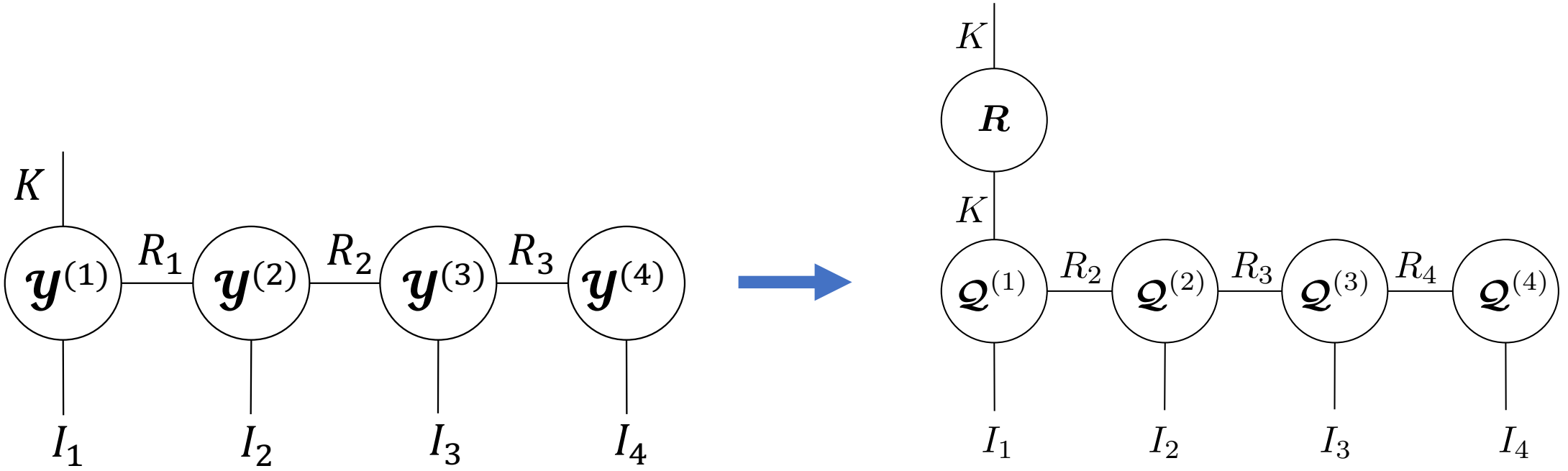


Tensor network randomized SVD



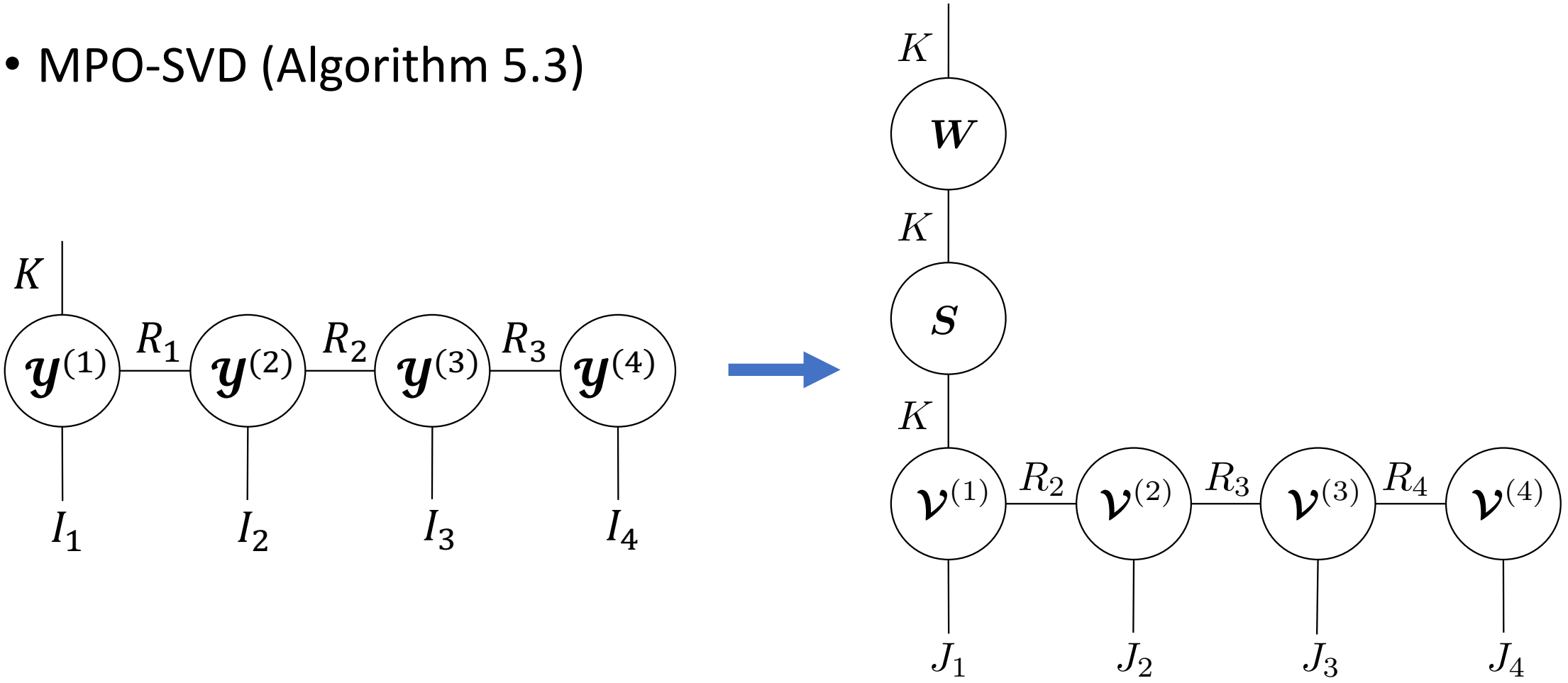
Tensor network randomized SVD

- MPO-QR (Algorithm 5.2)



Tensor network randomized SVD

- MPO-SVD (Algorithm 5.3)



Tensor network randomized SVD

ALGORITHM 5.4. *Randomized MPO-subspace iteration*
Input: *exponent* q, $\boldsymbol{A} \in \mathbb{R}^{I \times J}$ *and random matrix* $\boldsymbol{O} \in \mathbb{R}^{J \times K}$ *in MPO-form.*
Output: *MPO-tensors of* $\boldsymbol{Q} \in \mathbb{R}^{I \times K}$ *with* $\boldsymbol{Q}^T \boldsymbol{Q} = \boldsymbol{I}_K$.

$\boldsymbol{Y} \leftarrow \boldsymbol{AO}$
$\boldsymbol{Q} \leftarrow$ *Use Algorithm* 5.2 *on* $\boldsymbol{Y}$
for *i=1:q* ***do***
  $\boldsymbol{Y} \leftarrow \boldsymbol{A}^T \boldsymbol{Q}$
  $\boldsymbol{Q} \leftarrow$ *Use Algorithm* 5.2 *on* $\boldsymbol{Y}$ *with rounding*
  $\boldsymbol{Y} \leftarrow \boldsymbol{A}\,\boldsymbol{Q}$
  $\boldsymbol{Q} \leftarrow$ *Use Algorithm* 5.2 *on* $\boldsymbol{Y}$ *with rounding*
end for

Tensor network randomized SVD

ALGORITHM 5.5. *Tensor network randomized SVD*

Input: *matrix* $\boldsymbol{A} \in \mathbb{R}^{I \times J}$ *in MPO-form, target number* K*, oversampling parameter* s *and exponent* q*.*

Output: *approximate rank-*K *factorization* $\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$*, where* $\boldsymbol{U}, \boldsymbol{V}$ *are orthogonal and in MPO-form,* $\boldsymbol{S}$ *is diagonal and nonnegative.*

Generate an $J \times (K + s)$ *random matrix* $\boldsymbol{O}$ *in MPO-form using Lemma* 5.1

$\boldsymbol{Q} \leftarrow$ *Orthogonal basis for the range of* $(\boldsymbol{A}\boldsymbol{A}^T)^q \boldsymbol{A}\boldsymbol{O}$ *using Algorithm* 5.4

Compute $\boldsymbol{B} = \boldsymbol{Q}^T \boldsymbol{A}$ *according to subsection* 5.3

Compute the economical SVD $\boldsymbol{B} = \boldsymbol{W}\boldsymbol{S}\boldsymbol{V}^T$ *using Algorithm* 5.3

Compute $\boldsymbol{U} = \boldsymbol{Q}\boldsymbol{W}$ *as* $\boldsymbol{\mathcal{Q}}^{(1)} \times_3 \boldsymbol{W}^T$

Tensor network randomized SVD

ALGORITHM 5.6. *q-adaptive TNrSVD*

Input: *matrix* $\boldsymbol{A} \in \mathbb{R}^{I \times J}$ *in MPO-form, target number* K*, oversampling parameter* s*.*

Output: *approximate rank-*K *factorization* $\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$*, where* $\boldsymbol{U}, \boldsymbol{V}$ *are orthogonal and in MPO-form,* $\boldsymbol{S}$ *is diagonal and nonnegative.*

Generate an $J \times (K+s)$ *random matrix* $\boldsymbol{O}$ *in MPO-form using Lemma* 5.1
$\boldsymbol{Q} \leftarrow$ *Orthogonal basis for the range of* $\boldsymbol{A}\,\boldsymbol{O}$ *using Algorithm* 5.2
Compute $\boldsymbol{B} = \boldsymbol{Q}^T\,\boldsymbol{A}$ *according to subsection* 5.3
Compute the economical SVD $\boldsymbol{B} = \boldsymbol{W}\boldsymbol{S}\boldsymbol{V}^T$ *using Algorithm* 5.3
Compute $\boldsymbol{U} = \boldsymbol{Q}\,\boldsymbol{W}$ *as* $\boldsymbol{\mathcal{Q}}^{(1)} \times_3 \boldsymbol{W}^T$
while *Termination criterion not met* ***do***
  $\boldsymbol{Q} \leftarrow$ *Orthogonal basis for the range of* $(\boldsymbol{A}\boldsymbol{A}^T)\,\boldsymbol{Q}$
  Compute $\boldsymbol{B} = \boldsymbol{Q}^T\,\boldsymbol{A}$ *according to subsection* 5.3
  Compute the economical SVD $\boldsymbol{B} = \boldsymbol{W}\boldsymbol{S}\boldsymbol{V}^T$ *using Algorithm* 5.3
  Compute $\boldsymbol{U} = \boldsymbol{Q}\,\boldsymbol{W}$ *as* $\boldsymbol{\mathcal{Q}}^{(1)} \times_3 \boldsymbol{W}^T$
end while

Tensor network randomized SVD

- Termination Criterion for q-adaptive

$$\gamma := \max_{1 \leq i \leq K} \frac{|{\sigma_i^{(k)}}^2 - {\sigma_i^{(k-1)}}^2|}{{\sigma_1^{(k)}}^2}$$

- Where $\sigma_i^{(k)}$ denotes the *i*th singular value computed at the *k*th iteration.

Numerical experiments

- Fast matrix-to-MPO conversion

Test matrices from the SuiteSparse Matrix Collection [8].

Matrix	Dimensions	Dimension factorization
Erdos972	5488×5488	$7^3 \times 2^4$
lhr10c	10672×10672	$29 \times 23 \times 2^4$
delaunay_n14	16384×16384	128×2^7
g7jac100	29610×29610	$329 \times 5 \times 3^2 \times 2$
venkat01	62424×62424	$289 \times 3^3 \times 2^3$

Numerical experiments

Runtimes and relative errors for three different matrix-to-MPO methods.

Matrix	Runtime [seconds]			Relative error		
	Alg. 4.1	TT-SVD	TT-cross	Alg. 4.1	TT-SVD	TT-cross
Erdos972	0.649	12.25	14.92	0	8.11e-15	1.016
lhr10c	1.075	22.32	135.37	0	4.07e-15	0.926
delaunay_n14	0.171	1302.2	18.77	0	8.45e-15	1.141
g7jac100	0.903	422.34	716.31	0	4.67e-11	0.791
venkat01	1.025	NA	NA	0	NA	NA

- Algorithm 4.1 always results in an exact representation.
- Applying the TT-SVD and TT-cross methods on the venkat01 matrix is not possible due to insufficient memory (64 GB).

Numerical experiments

- Comparison with ALS-SVD and MALS-SVD

Hilbert matrix, $\boldsymbol{H}$ of size $2^N \times 2^N$ is a symmetric matrix with entries:

$$\boldsymbol{H}(i,j) = (i+j-1)^{-1}, i, j = 1, 2, \ldots, 2^N$$

The submatrix, $\boldsymbol{A}$ of size $2^N \times 2^{N-1}$ is considered with $10 < N < 50$

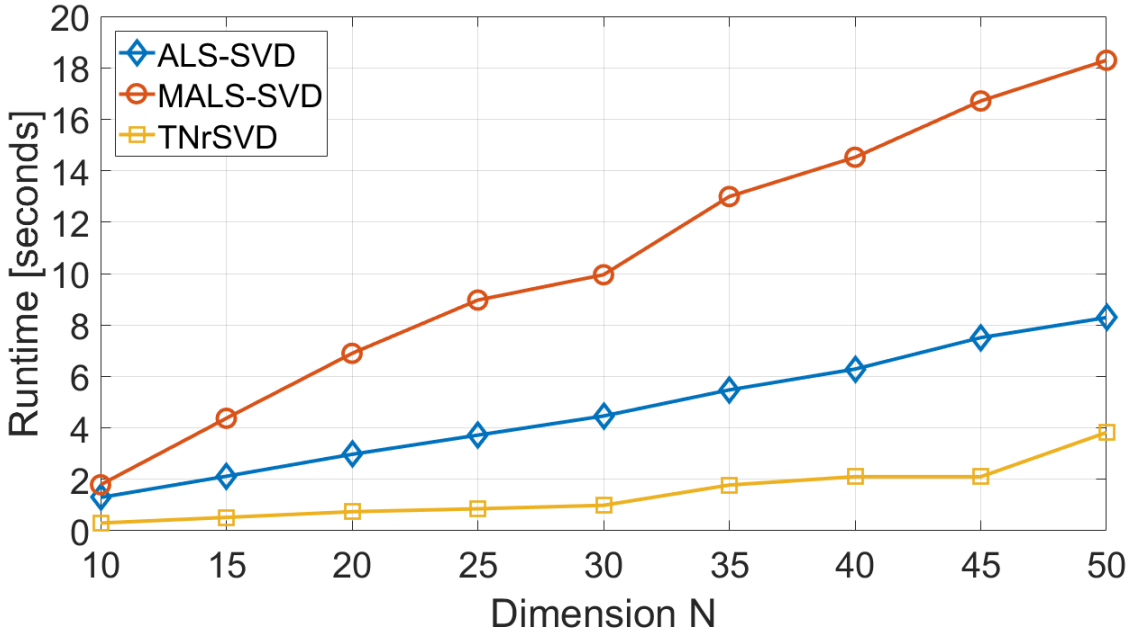


FIG. 6.3. *Runtimes for computing rank-16 approximations of* $2^N \times 2^{N-1}$ *Hilbert matrices for* $10 \leq N \leq 50$.

Numerical experiments

- Random matrix with prescribed singular values

Considering a matrix with 50 prescribed singular values 0.5^k, $(\mathrm{k} = 0, \ldots, 49)$ and random left and right singular vectors of size $2^N \times 50$, where N ranges from 10 to 50.

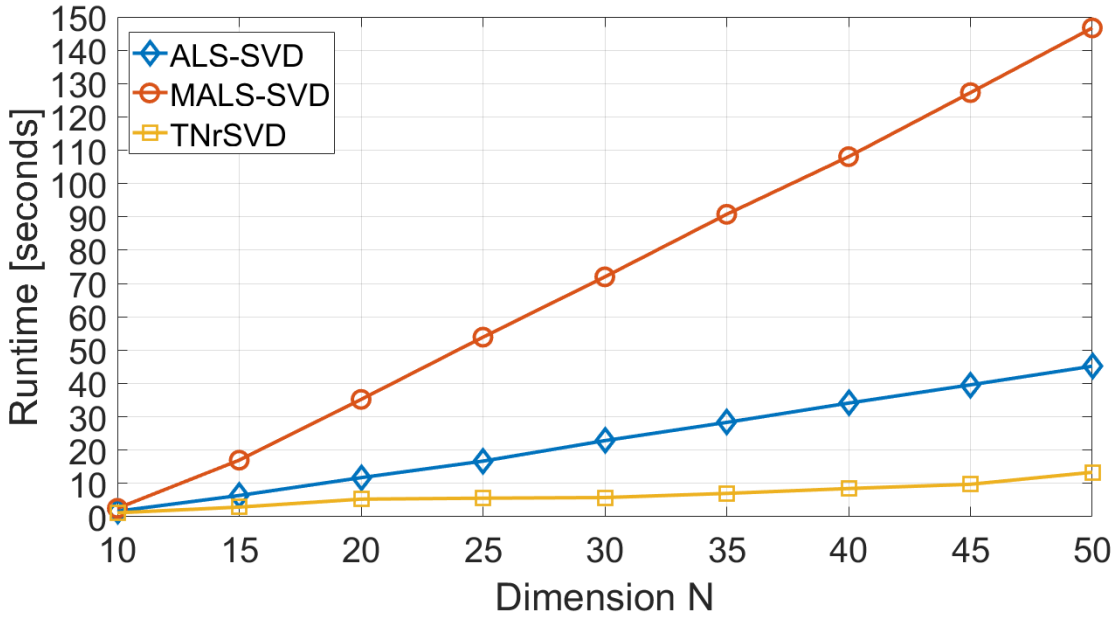


FIG. 6.4. *Runtimes for computing* rank-50 *decompositions of* $2^N \times 2^N$ *random matrices with prescribed singular values for* $10 \leq N \leq 50$.