ONE-DIMENSIONAL TENSOR NETWORK RECOVERY*

ZIANG CHEN†, JIANFENG LU‡, AND ANRU ZHANG§

Abstract. We study the recovery of the underlying graphs or permutations for tensors in the tensor ring or tensor train format. Our proposed algorithms compare the matricization ranks after down-sampling, whose complexity is $O(d \log d)$ for dth-order tensors. We prove that our algorithms can almost surely recover the correct graph or permutation when tensor entries can be observed without noise. We further establish the robustness of our algorithms against observational noise. The theoretical results are validated by numerical experiments.

Key words. tensor ring, tensor train, permutation recovery, almost-sure correctness, robustness against noise

MSC codes. 65C99, 15A69

DOI. 10.1137/23M159888X

1. Introduction. Tensors, or multidimensional arrays, are powerful in encoding and computing high-order information and have a wide range of applications in natural sciences and scientific computing. Since the storage and associated computational complexity of a tensor can be exponential in its order, many tensor formats are developed to reduce the storage and computation cost, e.g., the Tucker format [36] and tensor networks [25].

Tensor network models contract low-order tensors with respect to an underlying graph to form high-order tensors. They are widely used in many fields, such as quantum many-body physics [26, 25, 44, 30, 38, 12] and machine learning/data science [33, 7, 8, 6, 32, 28]. Among all tensor network structures, the one-dimensional network attracts the most attention due to its practical and efficient numerical performance. Such a structure represents high-order tensors using a group of 3-tensors ordered as a loop or a path and is known as matrix product states [30, 25] in physics and tensor ring (TR) [45], as well as the tensor train (TT) [27], in mathematics and numerical analysis, and thus, we adopt the name convention in this paper. The other tensor network formats include tree tensor networks [31], projected entangled pair states [25, 37], etc.

TR/TT decomposition [45, 27] is the problem to compute the TR/TT representation of given a high-order tensor and has many applications, such as tensor completion [43, 42, 2, 39], tensor SVD [46], and complexity reduction of neural network layers [40]. This direction has been well investigated from both algorithmic and theoretical aspects; we refer the interested readers to [5, 15, 22, 17, 4].

  †Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (ziang@mit.edu).
  ‡Departments of Mathematics, Physics, and Chemistry, Duke University, Durham, NC 27708 USA (jianfeng@math.duke.edu).
  §Departments of Biostatistics & Bioinformatics and Computer Science, Duke University, Durham, NC 27710 USA (anru.zhang@duke.edu).

The performance of tensor network decomposition tasks relies heavily on the choice of the graph. One gets better performance if one employs the right underlying graph for a target tensor in some tensor network format. Thus, in fields such as machine learning, scientific computing, and theoretical chemistry, several works are devoted to constructing or discovering the correct underlying graph for tensor networks. An agglomerative approach is proposed in [1] to find the suitable tree structure for hierarchical tensor formats. In [18], the authors view the underlying graph as binary strings and do searching in the Hamming space. Since then, there have been other works on searching the graph structure or topology for tensor networks; see, e.g., [13, 24, 21]. More related to our setting, [20] studies the tensor network permutation search, i.e., the problem of finding the best one-to-one mapping from the tensor indices to the vertices of a given graph. The algorithm in [20] uses local sampling to iteratively minimize some loss functions whose minimizer corresponds to the best permutation in some sense. The result is further improved in [19] with less computational cost. In quantum chemistry and physics literature, one approach proposed to find better index ordering or graph topology is to minimize the entanglement; see, e.g., [23, 34, 14] and references therein. [10] proposes another strategy using the inversion symmetry property of singular values.

In this work, we consider the recovery of the underlying graph for the TR and TT formats. More explicitly, for a given high-order tensor that is assumed to be of the TR or TT format, we aim to design efficient and reliable algorithms to recover the underlying graph, i.e., the permutation that maps the indices of the given tensor to correct ordering in the underlying loop or path.

We propose polynomial-time complexity algorithms with theoretical guarantees to solve the tensor indices permutation problem for the TR/TT format. To the best of our knowledge, this is the first approach with complete rigorous analysis including clear complexity bounds, almost-sure correctness in the noiseless case, and robustness theorems against the observation error.

The rest of this paper will be organized as follows. In section 2, after introducing tensor notations and the TR/TT formats, we define the task of recovering the underlying graph. The algorithms will be described in section 3 and analyzed rigorously in section 4. Section 5 contains some numerical experiments, and the whole paper will be concluded and discussed in section 6.

2. Problem statement. In this section, we first recall the definitions and notations of two one-dimensional tensor networks: TR and TT. Then, we state the one-dimensional tensor network recovery problem and our goals. For the rest of the paper, S_d is the permutation group on $\{1, 2, \ldots, d\}$; for a matrix M and a positive integer R, let $\sigma_R(M)$ be the Rth largest singular value of M.

2.1. Tensors. In this paper, tensors are referred to as multidimensional arrays. We focus on the real-valued tensors, while the developed results can be directly extended to the complex value cases. Specifically, a d-tensor, or a tensor with order d, is some $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ with entries being $\mathbf{X}(x_1, x_2, \ldots, x_d) \in \mathbb{R}$, $1 \leq x_i \leq n_i$, $1 \leq i \leq d$. We call $\vec{n} = (n_1, n_2, \ldots, n_d)$ the physical or external dimension of the tensor $\mathbf{X}$. Examples of tensors include vectors (1-tensors) and matrices (2-tensors). We introduce two operations on tensors below.

Tensor product: Let $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ be a d-tensor, and let $\mathbf{Y} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_{d'}}$ be a d'-tensor. The tensor product of $\mathbf{X}$ and $\mathbf{Y}$, denoted as $\mathbf{X} \otimes \mathbf{Y}$, is a $(d+d')$-tensor in $\mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d \times m_1 \times m_2 \times \cdots \times m_{d'}}$ defined as

$$\mathbf{X} \otimes \mathbf{Y}(x_1, x_2 \ldots, x_d, y_1, y_2, \ldots, y_{d'}) = \mathbf{X}(x_1, x_2, \ldots, x_d)\mathbf{Y}(y_1, y_2, \ldots, y_{d'})$$

for $1 \le x_i \le n_i$, $1 \le y_{i'} \le m_j$, $1 \le i \le d$, and $1 \le i' \le d'$.

Contraction: Let $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ be a d-tensor, and let $\mathbf{Y} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_{d'}}$ be a d'-tensor. Suppose that $n_i = m_{i'}$ for some $i \in \{1, 2, \ldots, d\}$ and $i' \in \{1, 2, \ldots, d'\}$. Then, $\mathbf{X}$ and $\mathbf{Y}$ can be contracted on the two indices with the same dimension, and the resulting tensor $\mathbf{Z} \in \mathbb{R}^{n_1 \times \cdots \times n_{i-1} \times n_{i+1} \times \cdots \times n_d \times m_1 \times \cdots \times m_{i'-1} \times m_{i'+1} \times \cdots \times m_{d'}}$ is a $(d + d' - 2)$-tensor defined via

$$\mathbf{Z}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d, y_1, \ldots, y_{i'-1}, y_{i'+1}, \ldots, y_{d'})$$
$$= \sum_{z=1}^{n_i} \mathbf{X}(x_1, \ldots, x_{i-1}, z, x_{i+1}, \ldots, x_d)\mathbf{Y}(y_1, \ldots, y_{i'-1}, z, y_{i'+1}, \ldots, y_{d'}).$$

The product of two matrices is a specific example of tensor contraction. Contraction can also be defined on a single tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ as long as $n_i = n_{i'}$ for some $1 \le i < i' \le d$, which results in $\mathbf{Z} \in \mathbb{R}^{n_1 \times \cdots \times n_{i-1} \times n_{i+1} \times \cdots \times n_{i'-1} \times n_{i'+1} \times \cdots \times n_d}$ with

$$\mathbf{Z}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{i'-1}, x_{i'+1}, \ldots, x_d)$$
$$= \sum_{z=1}^{n_i} \mathbf{X}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{i'-1}, x_{i'+1}, \ldots, x_d).$$

In particular, trace $\mathrm{tr}(\mathbf{X})$ of a square matrix $\mathbf{X} \in \mathbb{R}^{n \times n}$ is a contraction.

2.2. One-dimensional tensor networks. The tensor network model renders a powerful method to parameterize a high-order tensor by contraction operation on a collection of low-order tensors. The readers are referred to [25] for discussions on general tensor network models. In this work, we focus on two prominent one-dimensional tensor network models: TR and TT.

2.2.1. TR structure. Let $d \in \mathbb{N}_+$ be a fixed order. For the convenience of presentation, all index symbols on the TR are in a sense of mod d, e.g., $\mathbf{u}^{[d+1]} := \mathbf{u}^{[1]}$, $r_{d+1} = r_1$, $k_{d+1} = k_1$. A TR consists of d 3-tensors connected through contraction, as in Figure 1, where each node has three edges and represents a 3-tensor. For any $i \in \{1, 2, \ldots, d\}$, $\mathbf{u}^{[i]}$ shares a common edge with $\mathbf{u}^{[i+1]}$, meaning that a contraction is applied on $\mathbf{u}^{[i]}$ and $\mathbf{u}^{[i+1]}$. To define rigorously, given the external/physical dimension $\vec{n} = (n_1, n_2, \ldots, n_d)$ and internal/bond dimension $\vec{r} = (r_1, r_2, \ldots, r_d)$, any collection of 3-tensors

$$\vec{\mathbf{u}} = \left(\mathbf{u}^{[1]}, \mathbf{u}^{[2]}, \ldots, \mathbf{u}^{[d]}\right) \in \mathcal{U}_{\vec{r}, \vec{n}}^d = \bigoplus_{i=1}^{d} \mathbb{R}^{r_i \times n_i \times r_{i+1}}$$

yields a d-tensor in the TR format:

$$(2.1) \quad \varphi(\vec{\mathbf{u}})(x_1, x_2, \ldots, x_d) = \mathrm{tr}\left(\mathbf{u}^{[1]}(x_1)\mathbf{u}^{[2]}(x_2) \cdots \mathbf{u}^{[d]}(x_d)\right), \quad 1 \le x_i \le n_i, 1 \le i \le d,$$

where $\mathbf{u}^{[i]}(x_i) := \mathbf{u}^{[i]}(:, x_i, :) \in \mathbb{R}^{r_i \times r_{i+1}}$. Recall that matrix multiplication and trace are both tensor contractions, and (2.1) is well-defined since the third dimension of $\mathbf{u}^{[i]}$ coincides with the first dimension of $\mathbf{u}^{[i+1]}$, both equal to r_{i+1}, for $i = 1, 2, \ldots, d$. An equivalent way to describe $\varphi(\vec{\mathbf{u}})$ is via tensor product:

$$(2.2) \qquad \varphi(\vec{\mathbf{u}}) = \sum_{1 \le k_i \le r_i, \ 1 \le i \le d} \mathbf{u}_{k_1, k_2}^{[1]} \otimes \mathbf{u}_{k_2, k_3}^{[2]} \otimes \cdots \otimes \mathbf{u}_{k_d, k_1}^{[d]},$$

where $\mathbf{u}_{k_i, k_{i+1}}^{[i]} = \mathbf{u}^{[i]}(k_i, :, k_{i+1}) \in \mathbb{R}^{n_i}$, $1 \le k_i \le r_i$, $1 \le i \le d$.
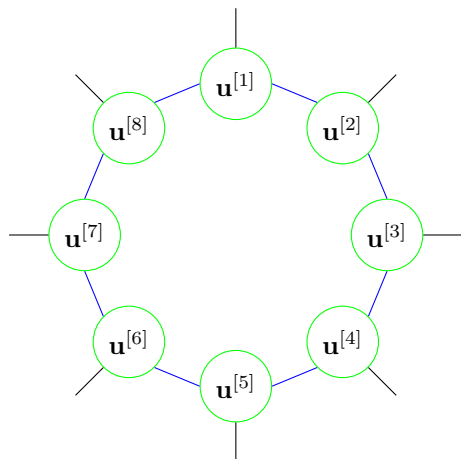
FIG. 1. *TR format.*
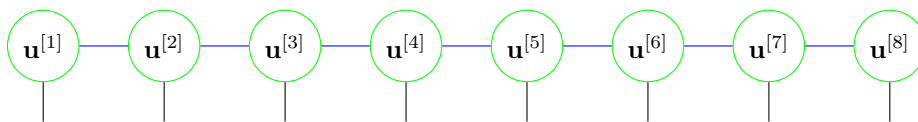
FIG. 2. *TT format.*

2.2.2. TT structure. The TT format is a special case of TR format where $r_1 = 1$ is fixed, and hence, the contraction between $\mathbf{u}^{[d]}$ and $\mathbf{u}^{[1]}$ is trivial. The TT model is shown in Figure 2, where there is no edge connecting $\mathbf{u}^{[d]}$ and $\mathbf{u}^{[1]}$ due to the trivial contraction. The expression of entries of $\varphi(\vec{\mathbf{u}})$ (2.1) becomes
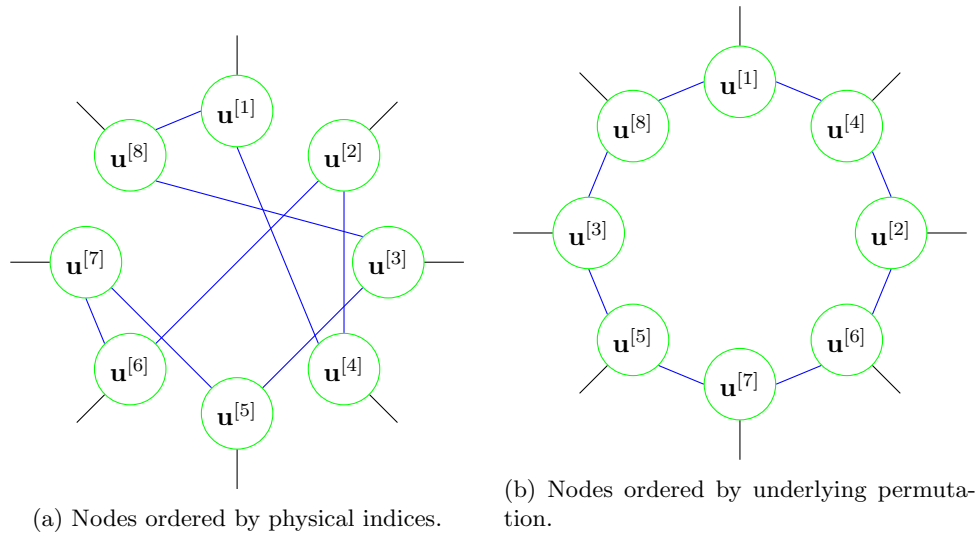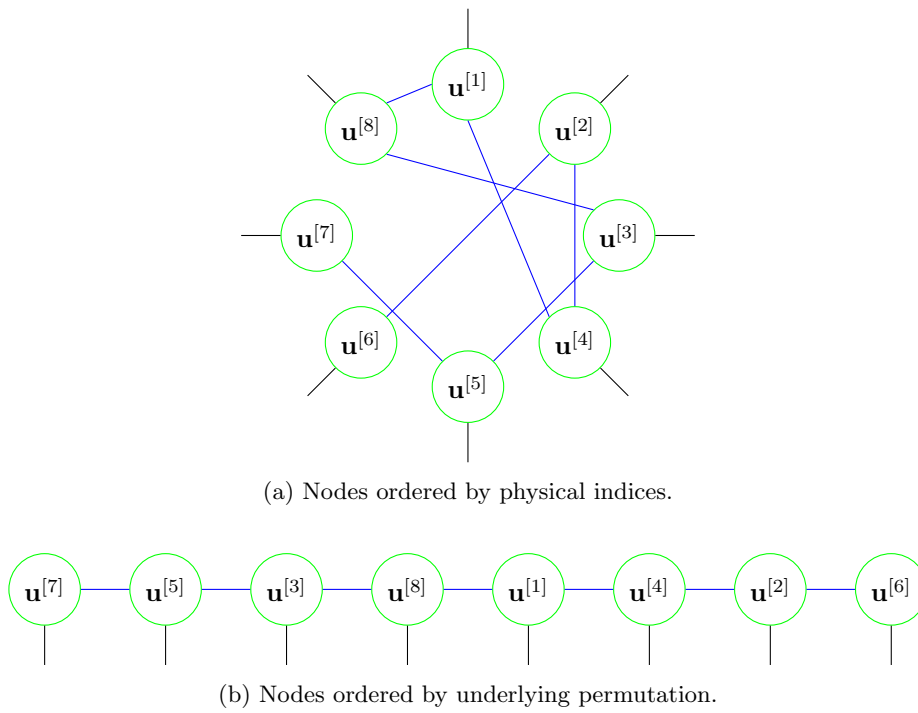
$$\varphi(\vec{\mathbf{u}})(x_1, x_2, \ldots, x_d) = \mathbf{u}^{[1]}(x_1)\mathbf{u}^{[2]}(x_2) \cdots \mathbf{u}^{[d]}(x_d).$$

If we denote $\mathbf{u}_{k_2}^{[1]} = \mathbf{u}^{[1]}(1, :, k_2) \in \mathbb{R}^{n_1}$ for $1 \le k_2 \le r_2$ and $\mathbf{u}_{k_d}^{[d]} = \mathbf{u}^{[d]}(k_d, :, 1) \in \mathbb{R}^{n_d}$ for $1 \le k_d \le r_d$, then (2.2) can be rewritten as

$$\varphi(\vec{\mathbf{u}}) = \sum_{1 \le k_j \le r_j,\ 2 \le j \le d} \mathbf{u}_{k_2}^{[1]} \otimes \mathbf{u}_{k_2,k_3}^{[2]} \otimes \cdots \otimes \mathbf{u}_{k_{d-1},k_d}^{[2]} \otimes \mathbf{u}_{k_d}^{[d]}.$$

2.3. Recovery of the underlying graph. In Figures 1 and 2, the construction of the TR and TT formats requires the prior knowledge of what pairs of 3-tensors are contracted, or, in other words, the underlying graph of the tensor network structure (see the blue parts in Figures 1 and 2). Note that, for the TR format, the graph is simply a loop/ring (hence named the tensor ring) with d vertices and d edges, where each vertex corresponds to a 3-tensor and is associated with two edges. For the TT format, the graph is a path of length $d - 1$ that visits all d vertices.

However, such prior knowledge is often inaccessible. In this work, we assume that the underlying graph is unknown. Since the graph for the TR or TT format is just a loop or a path, all information carried by the graph is the ordering of the vertices. This can be specified as a permutation $\tau \in S_d$, which then fixes the neighbors (vertex with distance 1) of each vertex. For example, one can choose $\tau = \left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 4 & 2 & 6 & 7 & 5 & 3 & 8 \end{smallmatrix}\right)$ for the TR format in Figure 3 and $\tau = \left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 7 & 5 & 3 & 8 & 1 & 4 & 2 & 6 \end{smallmatrix}\right)$ for the TT format in Figure 4.

(a) Nodes ordered by physical indices.

(b) Nodes ordered by underlying permutation.

FIG. 3. *TR format with permutation.*

(a) Nodes ordered by physical indices.

(b) Nodes ordered by underlying permutation.

FIG. 4. *TT format.*

We introduce the rigorous notations for the TR and TT formats with permutation as follows.

2.3.1. TR structure with permutation. Given permutation $\tau \in S_d$, external dimension $\vec{n} = (n_1, n_2, \ldots, n_d)$, and internal dimension $\vec{r} = (r_1, r_2, \ldots, r_d)$, denote

$$\mathcal{U}^d_{\vec{r},\vec{n},\tau} = \bigoplus_{i=1}^d \mathbb{R}^{r_{\tau^{-1}(i)} \times n_i \times r_{\tau^{-1}(i)+1}}.$$

Note that the indices/inputs for τ, r, etc., are in the sense of mod d, e.g., $\tau(d+1) = \tau(1), r_{d+1} = r_1$. An element in $\mathcal{U}^d_{\vec{r},\vec{n},\tau}$ can be written as a collection of 3-tensors:

$$\vec{\mathbf{u}} = \left(\mathbf{u}^{[1]}, \mathbf{u}^{[2]}, \dots, \mathbf{u}^{[d]}\right) \in \mathcal{U}^d_{\vec{r},\vec{n},\tau}, \quad \mathbf{u}^{[i]} \in \mathbb{R}^{r_{\tau^{-1}(i)} \times n_i \times r_{\tau^{-1}(i)+1}}, \; 1 \le i \le d.$$

Note that the third dimension of $\mathbf{u}^{[\tau(j)]}$ coincides with the first dimension of $\mathbf{u}^{[\tau(j+1)]}$, and they both equal r_{i+1}. This corresponds to the fact that, in the graph, there is an edge connecting the vertices representing $\mathbf{u}^{[\tau(j)]}$ and $\mathbf{u}^{[\tau(j+1)]}$ that can be contracted.

Denote

$$\mathcal{U}^d_{\vec{r},\vec{n}} = \bigcup_{\tau \in S_d} \mathcal{U}^d_{\vec{r},\vec{n},\tau} \times \{\tau\}$$

that consists of all consistent groups of 3-tensors as well as the permutation. The map that constructs a d-tensor with TR format is given by

$$\varphi : \mathcal{U}^d_{\vec{r},\vec{n}} \to \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$$
$$(\vec{\mathbf{u}}, \tau) \mapsto \varphi(\vec{\mathbf{u}}, \tau)$$

with entries of $\varphi(\vec{\mathbf{u}}, \tau)$ being

$$(2.3) \qquad \varphi(\vec{\mathbf{u}}, \tau)(x_1, x_2, \dots, x_d) = \mathrm{tr}\left(\mathbf{u}^{[\tau(1)]}(x_{\tau(1)}) \mathbf{u}^{[\tau(2)]}(x_{\tau(2)}) \cdots \mathbf{u}^{[\tau(d)]}(x_{\tau(d)})\right)$$

for $1 \le x_i \le n_i$, $1 \le i \le d$, where $\mathbf{u}^{[i]}(x_i) := \mathbf{u}^{[i]}(:, x_i, :) \in \mathbb{R}^{r_{\tau^{-1}(i)} \times r_{\tau^{-1}(i)+1}}$. One can also define $\varphi(\vec{\mathbf{u}}, \tau)$ using the tensor product:

$$(2.4) \qquad \varphi(\vec{\mathbf{u}}, \tau) = \sum_{1 \le k_j \le r_j, \; 1 \le j \le d} \mathbf{u}^{[\tau(1)]}_{k_1,k_2} \otimes \mathbf{u}^{[\tau(2)]}_{k_2,k_3} \otimes \cdots \otimes \mathbf{u}^{[\tau(d)]}_{k_d,k_1},$$

where $\mathbf{u}^{[\tau(j)]}_{k_j,k_{j+1}} = \mathbf{u}^{[\tau(j)]}(k_j, :, k_{j+1}) \in \mathbb{R}^{n_{\tau(j)}}$, $1 \le k_j \le r_j$, $1 \le j \le d$, and $k_{d+1} := k_1$.

One observation is that different permutations may result in the same graph, i.e., the same pairs of adjacent 3-tensors. More explicitly, let $\alpha, \beta \in S_d$ be defined via $\alpha(j) = j + 1$ and $\beta(j) = d + 1 - j$ for $j = 1, 2, \dots, d$. Then, for any $\tau, \tau' \in S_d$, their underlying graphs are the same if and only if there exists $k \in \{0, 1, \dots, d-1\}$ and $\ell \in 0, 1$ such that $\tau' = \tau \circ \alpha^k \circ \beta^\ell$. In other words, one divides S_d into several equivalence classes, and permutations in the same class lead to the same graph. The equivalence class containing $\tau \in S_d$ is

$$(2.5) \qquad \mathcal{CS}^d_{\mathrm{TR}}(\tau) = \left\{\tau \circ \alpha^k \circ \beta^\ell : k \in \{0, 1, \dots, d-1\}, \; \ell \in 0, 1\right\}.$$

2.3.2. TT structure with permutation. The TT format can still be obtained from the TR format by fixing $r_1 = 1$, which modifies (2.3) into

$$\varphi(\vec{\mathbf{u}}, \tau)(x_1, x_2, \dots, x_d) = \mathbf{u}^{[\tau(1)]}(x_{\tau(1)}) \mathbf{u}^{[\tau(2)]}(x_{\tau(2)}) \cdots \mathbf{u}^{[\tau(d)]}(x_{\tau(d)}).$$

Rewrite (2.4) as

$$\varphi(\vec{\mathbf{u}}, \tau) = \sum_{1 \le k_j \le r_j, \; 2 \le j \le d} \mathbf{u}^{[\tau(1)]}_{k_2} \otimes \mathbf{u}^{[\tau(2)]}_{k_2,k_3} \otimes \cdots \otimes \mathbf{u}^{[\tau(d-1)]}_{k_{d-1},k_d} \otimes \mathbf{u}^{[\tau(d)]}_{k_d},$$

where $\mathbf{u}_{k_2}^{[\tau(1)]} = \mathbf{u}^{[\tau(1)]}(1,:,k_2) \in \mathbb{R}^{n_{\tau(1)}}$ for $1 \leq k_2 \leq r_2$ and $\mathbf{u}_{k_d}^{[\tau(d)]} = \mathbf{u}^{[\tau(d)]}(k_d,:,1) \in \mathbb{R}^{n_{\tau(d)}}$ for $1 \leq k_d \leq r_d$.

Similar to the TR case, the permutation corresponding to a graph or path for the TT format is not unique. The difference is that the equivalence class containing τ is

$$(2.6) \qquad \mathcal{C}S_{\mathrm{TT}}^d(\tau) = \{\tau, \tau \circ \beta\}.$$

2.3.3. Tasks and goals. In this work, we study how to infer the unknown underlying graph for TR and TT structures via observations of entries of the whole tensor $\mathbf{T}$. More specifically, we would like to find all pairs of 3-tensors for which the associated vertices are adjacent (connected via an edge), that are $(\mathbf{u}^{[\tau(j)]}, \mathbf{u}^{[\tau(j+1)]})$, $j = 1, 2, \ldots, d$ for TR format, and are $(\mathbf{u}^{[\tau(j)]}, \mathbf{u}^{[\tau(j+1)]})$, $j = 1, 2, \ldots, d-1$ for TT format. For simplicity, we say two 3-tensors are adjacent if their associated vertices are adjacent. Our goal is to design algorithms that are efficient both in sample complexity (number of observed tensor entries) and computational complexity. To sum up, the goal of this paper is the following:

> Given an oracle that can query (noiseless or noisy) entries of the whole tensor $\mathbf{T} = \varphi(\vec{\mathbf{u}}, \tau) \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ of the TR or TT format, where $(\vec{\mathbf{u}}, \tau) \in \mathcal{U}_{\vec{r}, \vec{n}}^d$ with $\vec{\mathbf{u}}$, τ, and $\vec{r}$ unknown, can we recover the underlying graph, i.e., find some permutation in the equivalence class $\mathcal{C}S_{\mathrm{TR}}^d(\tau)$ or $\mathcal{C}S_{\mathrm{TT}}^d(\tau)$? If yes, what is the complexity and the probability of success?

It is worth remarking that two permutations representing the same tensor $\mathbf{T}$ do not always belong to the same equivalence class. For example, if $\vec{r} = (1, \ldots, 1)$, then $\mathbf{T} = \varphi(\vec{\mathbf{u}}, \tau)$ can be represented by either the TR or TT format with respect to any permutation. To strengthen our argument, we prove in Appendix A that, under certain conditions, only permutations within the same equivalence class can yield the identical tensor $\mathbf{T}$. This result establishes the uniqueness of the correct permutation with respect to the equivalence relation.

A related question is how to identify a permutation that may not be in the equivalence class $\mathcal{C}S_{\mathrm{TR}}^d(\tau)$ or $\mathcal{C}S_{\mathrm{TT}}^d(\tau)$ but can represent the target tensor $\mathbf{T}$ with acceptable error or acceptably increased bond dimension. Although this topic of approximate recovery is important, it falls outside the scope of this paper, which concentrates on the precise recovery of the permutation.

3. Proposed algorithms. In this section, we introduce algorithms for recovering underlying graphs of the TR and TT formats as well as the intuition behind it. The main idea is divide and conquer: For determining the correct relative positions of all d indices on a loop (or a path), it suffices to determine the relative positions of any given 4 indices (for TR format) or 3 indices (for TT format). Therefore, we first design algorithms for these subproblems and then recover the whole graph based on solutions to the subproblems. The TR and TT cases will be covered in sections 3.1 and 3.2, respectively.

3.1. Recovery of the underlying loop for TR format. Given four indices in some order, a basic question is whether one could travel around the underlying loop such that the four indices are visited in exactly the same order as given. Note that four is the smallest number of indices such that the question is nontrivial. Thus, one can classify the order into "correct" or "incorrect" based on the answer to the question. It is clear that the classification is independent of rotations and reflections. Based on this observation, we make the following definitions.

DEFINITION 3.1. *For the TR format with $\tau \in S_d$ being the underlying permutation and four different indices $i_1, i_2, i_3, i_4 \in \{1, 2, \ldots, d\}$, we say that (i_1, i_2, i_3, i_4) is of the* correct order *with respect to τ if there exists some $s \in \{1, 2, 3, 4\}$ such that $\tau^{-1}(i_s) < \tau^{-1}(i_{s+1}) < \tau^{-1}(i_{s+2}) < \tau^{-1}(i_{s+3})$ or $\tau^{-1}(i_s) > \tau^{-1}(i_{s+1}) > \tau^{-1}(i_{s+2}) > \tau^{-1}(i_{s+3})$, where subscript of $i.$ is understood via mod 4.*

Figure 3 illustrates an example: $(1, 2, 6, 3)$ is of the correct order, while $(1, 2, 3, 6)$ is not. For identifying the underlying loop, it suffices to consider whether four indices are of the correct order with respect to τ due to the following proposition.

PROPOSITION 3.2. *Let $\tau, \tau' \in S_d$. $\tau' \in \mathcal{CS}_{TR}^d(\tau)$ if and only if, for any $1 \le j_1 < j_2 < j_3 < j_4 \le d$, $(\tau'(j_1), \tau'(j_2), \tau'(j_3), \tau'(j_4))$ is of the correct order with respect to τ.*

The proof of 3.2 is straightforward and omitted.

Next, we focus on the problem of determining the correct order of four specific indices $i_1, i_2, i_3, i_4 \in \{1, 2, \ldots, d\}$. The main idea is to construct three matrices and compare their ranks or singular values. More specifically, the three matrices are $M_{(i_1,i_2),(i_3,i_4)} \in \mathbb{R}^{(n_{i_1} n_{i_2}) \times (n_{i_3} n_{i_4})}$, $M_{(i_1,i_3),(i_2,i_4)} \in \mathbb{R}^{(n_{i_1} n_{i_3}) \times (n_{i_2} n_{i_4})}$, and $M_{(i_1,i_4),(i_2,i_3)} \in \mathbb{R}^{(n_{i_1} n_{i_4}) \times (n_{i_2} n_{i_3})}$ defined via matricization of the tensor:

$$
\begin{aligned}
& M_{(i_1,i_2),(i_3,i_4)}((x_{i_1}, x_{i_2}), (x_{i_3}, x_{i_4})) \\
(3.1) \qquad & = \varphi(\vec{\mathbf{u}}, \tau)(y_1, \ldots, y_{i'_1-1}, x_{i'_1}, y_{i'_1+1}, \ldots, y_{i'_2-1}, x_{i'_2}, \\
& \qquad\qquad y_{i'_2+1}, \ldots, y_{i'_3-1}, x_{i'_3}, y_{i'_3+1}, \ldots, y_{i'_4-1}, x_{i'_4}, y_{i'_4+1}, \ldots, y_d),
\end{aligned}
$$

$$
(3.2) \quad M_{(i_1,i_3),(i_2,i_4)}((x_{i_1}, x_{i_3}), (x_{i_2}, x_{i_4})) = M_{(i_1,i_2),(i_3,i_4)}((x_{i_1}, x_{i_2}), (x_{i_3}, x_{i_4})),
$$

and

$$
(3.3) \qquad M_{(i_1,i_4),(i_2,i_3)}((x_{i_1}, x_{i_4}), (x_{i_2}, x_{i_3})) = M_{(i_1,i_2),(i_3,i_4)}((x_{i_1}, x_{i_2}), (x_{i_3}, x_{i_4}))
$$

for $1 \le x_{i_s} \le n_{i_s}$, $1 \le s \le 4$, where $\{i'_1, i'_2, i'_3, i'_4\} = \{i_1, i_2, i_3, i_4\}$, $1 \le i'_1 < i'_2 < i'_3 < i'_4 \le d$, and $y_i \in \{1, 2, \ldots, n_i\}$ is a fixed index for each $i \in \{1, 2, \ldots, d\} \setminus \{i_1, i_2, i_3, i_4\}$. These three matrices can be constructed by calling the oracle that returns entries of $\mathbf{T} = \varphi(\vec{\mathbf{u}}, \tau)$. Intuitively, if (i_1, i_2, i_3, i_4) is of the correct order, $r_1 = r_2 = \cdots = r_d = R$, and $\vec{n}$ is large enough, then $\mathrm{rank}(M_{(i_1,i_3),(i_2,i_4)}) = R^4$, while $\mathrm{rank}(M_{(i_1,i_2),(i_3,i_4)})$ and $\mathrm{rank}(M_{(i_1,i_4),(i_2,i_3)})$ are at most R^2, as shown in Figure 5. This leads to the design of Algorithm 3.1. An input R, ideally an approximation of the bond dimension of the TR structure, is required in Algorithm 3.1. We will discuss the selection of R later.

Remark 3.3. In the noiseless case, where entries of $\mathbf{T} = \varphi(\vec{\mathbf{u}}, \tau)$ can be observed exactly, noticing (4.4), one can replace the comparison of the R^4th largest singular values of $M_{(i_1,i_2),(i_3,i_4)}$, $M_{(i_1,i_3),(i_2,i_4)}$, and $M_{(i_1,i_4),(i_2,i_3)}$ by the comparison of the ranks of these three matrices. Then, Algorithm 3.1 can be implemented even if the positive integer R approximating the bond dimension is unknown.

Now, we consider our goal of obtaining some $\tau' \in \mathcal{CS}_{\mathrm{TR}}^d(\tau)$. By Proposition 3.2, it suffices to find some $\tau' \in S_d$ such that $(\tau'(j_1), \tau'(j_2), \tau'(j_3), \tau'(j_4))$ is of the correct order with respect to τ. Note that Algorithm 3.1 can return the correct order of any four indices that fits the requirement on the desired τ'. So, the idea is to apply Algorithm 3.1 repeatedly and to extend the sequence of indices in the correct order. We make this idea more precise in Definition 3.4.

DEFINITION 3.4. *For the TR format, let $\tau \in S_d$, and let $(i_1, i_2, \ldots, i_t)$ be a permutation of t different indices in $\{1, 2, \ldots, d\}$, $4 \le t \le d$. Then, $(i_1, i_2, \ldots, i_t)$ is said to be* consistent *with τ if $(i_{j_1}, i_{j_2}, i_{j_3}, i_{j_4})$ is of the correct order with respect to τ for any $1 \le j_1 < j_2 < j_3 < j_4 \le t$.*

(a) $M_{(i_1,i_2),(i_3,i_4)}$.

(b) $M_{(i_1,i_4),(i_2,i_3)}$.
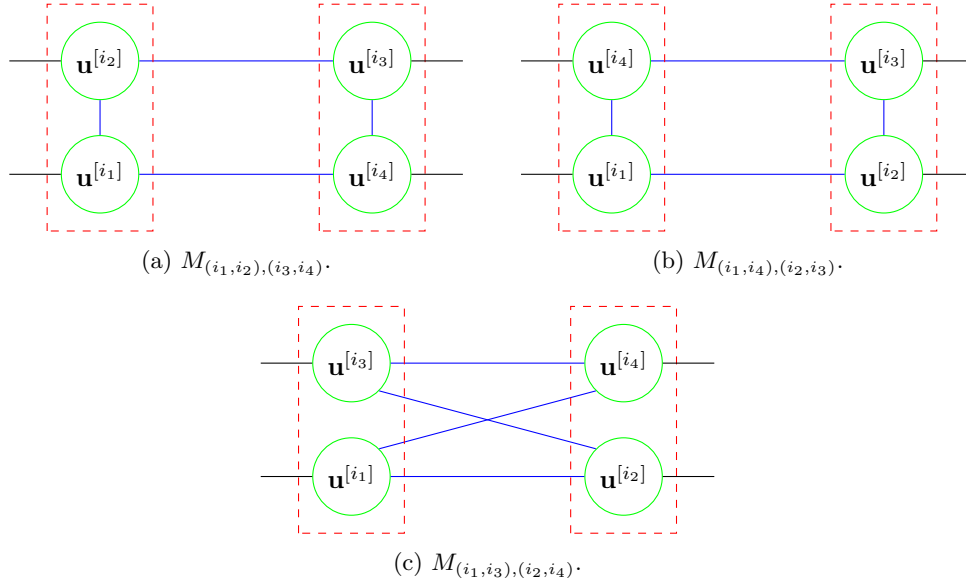
(c) $M_{(i_1,i_3),(i_2,i_4)}$.

FIG. 5. *Intuition of Algorithm* 3.1: *When* (i_1,i_2,i_3,i_4) *is of the correct order, the rank of* $M_{(i_1,i_2),(i_3,i_4)}$ *and* $M_{(i_1,i_4),(i_2,i_3)}$ *is at most* R^2, *while the rank of* $M_{(i_1,i_3),(i_2,i_4)}$ *is generically at least* R^4. *The reason is that the number of blue lines connecting red boxes is two in* (a) *and* (b) *but four in* (c), *where red boxes group the column indices in the matricization.*

Algorithm 3.1. Determine the order of four indices for the TR format.

1: **Input:** Four indices $i_1, i_2, i_3, i_4 \in \{1, 2, \ldots, d\}$, $R \in \mathbb{N}_+$ approximating the bond dimension, and an oracle computing entries of $\mathbf{T} = \varphi(\vec{\mathbf{u}}, \tau)$.

2: Sample $y_i \in \{1, 2, \ldots, n_i\}$ for $i \in \{1, 2, \ldots, d\} \backslash \{i_1, i_2, i_3, i_4\}$.

3: Construct matrices $M_{(i_1,i_2),(i_3,i_4)}$, $M_{(i_1,i_3),(i_2,i_4)}$, and $M_{(i_1,i_4),(i_2,i_3)}$ via (3.1), (3.2), and (3.3).

4: **if** $\sigma_{R^4}(M_{(i_1,i_2),(i_3,i_4)}) > \max\{\sigma_{R^4}(M_{(i_1,i_3),(i_2,i_4)}), \sigma_{R^4}(M_{(i_1,i_4),(i_2,i_3)})\}$ **then**

5:    Return (i_1, i_3, i_2, i_4).

6: **end if**

7: **if** $\sigma_{R^4}(M_{(i_1,i_3),(i_2,i_4)}) > \max\{\sigma_{R^4}(M_{(i_1,i_2),(i_3,i_4)}), \sigma_{R^4}(M_{(i_1,i_4),(i_2,i_3)})\}$ **then**

8:    Return (i_1, i_2, i_3, i_4).

9: **end if**

10: **if** $\sigma_{R^4}(M_{(i_1,i_4),(i_2,i_3)}) > \max\{\sigma_{R^4}(M_{(i_1,i_2),(i_3,i_4)}), \sigma_{R^4}(M_{(i_1,i_3),(i_2,i_4)})\}$ **then**

11:    Return (i_1, i_2, i_4, i_3).

12: **end if**

Suppose that $4 \leq t < d$ and we already have a permutation of $(1, 2, \ldots, t)$, say, $(i_1, i_2, \ldots, i_t)$, that is consistent with the correct complete order τ. We then aim to insert $(t + 1)$ into the right position in $(i_1, i_2, \ldots, i_t)$ that maintains the consistency with τ. For any $1 \leq j_1 < j_2 < j_3 \leq t$, one can run Algorithm 3.1 for $t + 1$, i_{j_1}, i_{j_2}, and i_{j_3} to determine where $t + 1$ should be inserted: 1) between i_{j_1} and i_{j_2}, 2) between i_{j_2} and i_{j_3}, or 3) before i_{j_1} or after i_{j_3}. Then, one can apply the technique similar to binary search to find the correct position of $t + 1$ by running Algorithm 3.1 for at most $O(\log t)$ times and hence obtaining a permutation of $(1, 2, \ldots, t + 1)$, say, $(i_1, i_2, \ldots, i_{t+1})$, that is still consistent with τ. The whole procedure is outlined in Algorithm 3.2.

Algorithm 3.2. Recover the underlying graph for the TR format.

1: **Input:** $R \in \mathbb{N}_+$ approximating the bond dimension and a oracle computing entries of $\mathbf{T} = \varphi(\vec{\mathbf{u}}, \tau)$.
2: Let (i_1, i_2, i_3, i_4) be the output of Algorithm 3.1 with the input four indices being $1, 2, 3, 4$.
3: **for** $t = 4 : d - 1$ **do**
4:    $j_{\min} = 1$, $j_{\max} = t + 1$. {Use the idea of binary search to find a location in $(i_1, i_2, \ldots, i_t)$ such that inserting $t + 1$ maintains the consistency with τ.}
5:    **while** $j_{\max} - j_{\min} \geq 2$ **do**
6:      **if** $j_{\max} - j_{\min} \geq 3$ **then**
7:        $j_1 = j_{\min}$, $j_2 = j_{\min} + \lfloor \frac{1}{3} \cdot (j_{\max} - j_{\min}) \rfloor$, $j_3 = j_{\min} + \lfloor \frac{2}{3} \cdot (j_{\max} - j_{\min}) \rfloor$.
8:      **else if** $j_{\max} \leq t$ **then**
9:        $j_1 = j_{\min}$, $j_2 = j_{\min} + 1$, $j_3 = j_{\max}$.
10:      **else**
11:        $j_1 = 1$, $j_2 = t - 1$, $j_3 = t$.
12:      **end if**
13:      Run Algorithm 3.1 with input being $t + 1$, i_{j_1}, i_{j_2}, and i_{j_3}.
14:      **if** the output of Algorithm 3.1 is equivalent to $(i_{j_1}, t + 1, i_{j_2}, i_{j_3})$ **then**
15:        $j_{\min} = j_1$, $j_{\max} = j_2$.
16:      **else if** the output of Algorithm 3.1 is equivalent to $(i_{j_1}, i_{j_2}, t + 1, i_{j_3})$ **then**
17:        $j_{\min} = j_2$, $j_{\max} = j_3$.
18:      **else**
19:        $j_{\min} = j_3$.
20:      **end if**
21:    **end while**
22:    $(i_1, i_2, \ldots, i_{t+1}) = (i_1, i_2, \ldots, i_{j_{\min}}, t + 1, i_{j_{\max}}, i_{j_{\max}+1}, \ldots, i_t)$.
23: **end for**
24: Return $\tau' \in S_d$ with $\tau'(j) = i_j$, $1 \leq j \leq d$.

Remark 3.5. Algorithm 3.2 combines the partial order of four indices output by Algorithm 3.1 to recover the order of all d indices. To make the overall procedure more robust, one can run Algorithm 3.1 several times on each quadruplet $(t + 1, i_{j_1}, i_{j_2}, i_{j_3})$ and take the majority vote of the outputs to determine their order.

Algorithm 3.2 can be implemented by observing $O(d \log d \cdot n_{\max}^4)$ entries of $\mathbf{T} = \varphi(\vec{\mathbf{u}}, \tau)$ and computing the R^4th singular value of $O(d \log d)$ matrices of size no larger than $n_{\max}^2 \times n_{\max}^2$, where $n_{\max} = \max_{1 \leq i \leq d} n_i$. The total complexity of Algorithm 3.2 is $O(d \log d \cdot n_{\max}^6)$. (Recall that the computational complexity of SVD is $O(m_1 m_2 \cdot \min\{m_1, m_2\})$ for a $m_1 \times m_2$ matrix [35].) It is also possible to further reduce the computational complexity using randomized SVD (see, e.g., [11]), but we will not go into the details here.

3.2. Recovery of the underlying path for TT format. The idea for recovering the underlying graph of the TT format is similar to that of the TR format. The difference is that the rotational invariance of the TR format no longer holds for the TT format; i.e., $\alpha \in S_d$ defined via $\alpha(j) = j + 1$, $j \in \{1, 2, \ldots, d\}$ appears in (2.5), but not in (2.6). Therefore, the ordering of four indices in TT format has 12 different cases, much more complicated than in TR format. As an alternative, we consider the relative positions of three indices, which leads to the simplest nontrivial subproblems and is defined below with reflective invariance.

DEFINITION 3.6. *For the TT format with $\tau \in S_d$ being the underlying permutation and three different indices $i_1, i_2, i_3 \in \{1, 2, \ldots, d\}$, we say that (i_1, i_2, i_3) is of the correct order with respect to τ if $\tau^{-1}(i_1) < \tau^{-1}(i_2) < \tau^{-1}(i_3)$ or $\tau^{-1}(i_3) < \tau^{-1}(i_2) < \tau^{-1}(i_1)$.*

Similarly, the relative positions of all triples of indices provide all information about the underlying permutation up to reflection.

PROPOSITION 3.7. *Let $\tau, \tau' \in S_d$. $\tau' \in \mathcal{CS}_{TR}^d(\tau)$ if and only if, for any $1 \leq j_1 < j_2 < j_3 \leq d$, $(\tau'(j_1), \tau'(j_2), \tau'(j_3))$ is of the correct order with respect to τ.*

For any three indices i_1, i_2, i_3, determining their order can still be done by investigating three matrices obtained by down-sampling and reshaping/matricizing, analogous to the TR case: $M_{i_1,(i_2,i_3)} \in \mathbb{R}^{n_{i_1} \times (n_{i_2} n_{i_3})}$, $M_{i_2,(i_3,i_1)} \in \mathbb{R}^{n_{i_2} \times (n_{i_3} n_{i_1})}$, and $M_{i_3,(i_1,i_2)} \in \mathbb{R}^{n_{i_3} \times (n_{i_1} n_{i_2})}$ with entries being

$$
\begin{aligned}
M_{i_1,(i_2,i_3)}&(x_{i_1}, (x_{i_2}, x_{i_3})) \\
&= \tau(\vec{\mathbf{u}}, \tau)(y_1, \ldots, y_{i'_1 - 1}, x_{i'_1}, y_{i'_1 + 1}, \ldots, y_{i'_2 - 1}, \\
&\qquad x_{i'_2}, y_{i'_2 + 1}, \ldots, y_{i'_3 - 1}, x_{i'_3}, y_{i'_3 + 1}, \ldots, y_d),
\end{aligned} \tag{3.4}
$$

$$
M_{i_2,(i_3,i_1)}(x_{i_2}, (x_{i_3}, x_{i_1})) = M_{i_1,(i_2,i_3)}(x_{i_1}, (x_{i_2}, x_{i_3})), \tag{3.5}
$$

and

$$
M_{i_3,(i_1,i_2)}(x_{i_3}, (x_{i_1}, x_{i_2})) = M_{i_1,(i_2,i_3)}(x_{i_1}, (x_{i_2}, x_{i_3})) \tag{3.6}
$$

for $1 \leq x_{i_s} \leq n_{i_s}$, $1 \leq s \leq 3$, where $\{i'_1, i'_2, i'_3\} = \{i_1, i_2, i_3\}$, $1 \leq i'_1 < i'_2 < i'_3 \leq d$, and $y_i \in \{1, 2, \ldots, n_i\}$ is a fixed index for each $i \in \{1, 2, \ldots, d\} \backslash \{i_1, i_2, i_3\}$. Similar to the TR case, one can obtain some intuition by considering the case with $r_2 = r_3 = \cdots = r_d = R$ and sufficiently large $\vec{n}$: If (i_1, i_2, i_3) is of the correct order, then the ranks of $M_{i_1,(i_2,i_3)}$ and $M_{i_3,(i_1,i_2)}$ are both no larger than R, while the rank of $M_{i_2,(i_3,i_1)}$ could be R^2. This is shown in Figure 6. A more precise procedure is stated in Algorithm 3.3 that is based on $R \in \mathbb{N}_+$, which is an approximation to the bond dimension.

Similarly, Algorithm 3.3 can also be implemented without knowing R in the noiseless case. The idea of building a final algorithm based on Algorithm 3.3 is also similar to that of the TR format, and we would need the following definition.

DEFINITION 3.8. *Let $\tau \in S_d$, and let $(i_1, i_2, \ldots, i_t)$ be a permutation of t different indices in $\{1, 2, \ldots, d\}$, $3 \leq t \leq d$. Then, $(i_1, i_2, \ldots, i_t)$ is said to be consistent with τ if $(i_{j_1}, i_{j_2}, i_{j_3})$ is of the correct order with respect to τ for any $1 \leq j_1 < j_2 < j_3 \leq t$.*

Similar to the TR case, we may recover the whole path by inserting indices one by one. Suppose that, for $3 \leq t \leq d - 1$, $(1, 2, \ldots, t)$ is permuted to $(i_1, i_2, \ldots, i_t)$ that is consistent with τ, and one aims to insert $t + 1$ and keeps the consistency. For any $1 \leq j_1 < j_2 \leq t$, the output of Algorithm 3.3 with the three input indices being $t + 1$, i_{j_1}, and i_{j_2} indicates that whether $t + 1$ should be placed before i_{j_1}, between i_{j_1} and i_{j_2}, or after i_{j_2}. By using a similar idea as in the binary search, one could identify the correct position to insert $t + 1$ by calling Algorithm 3.3 for $O(\log n)$ times. Algorithm 3.4 displays the whole procedure.

Similar to Algorithm 3.2, Algorithm 3.4 also collects and combines partial information, and there are other approaches such as majority vote. In addition, the complexity of Algorithm 3.4 is also similar to that of Algorithm 3.2, which is polynomial in d. More specifically, implementing Algorithm 3.4 requires observing $O(d \log d \cdot n_{\max}^3)$ entries of $\mathbf{T} = \varphi(\vec{\mathbf{u}}, \tau)$ and computing the R^2th largest singular value of $O(d \log d)$ matrices whose sizes are at most $n_{\max} \times n_{\max}^2$ with $n_{\max} = \max_{1 \leq i \leq d} n_i$, leading to the overall complexity $O(d \log d \cdot n_{\max}^4)$.
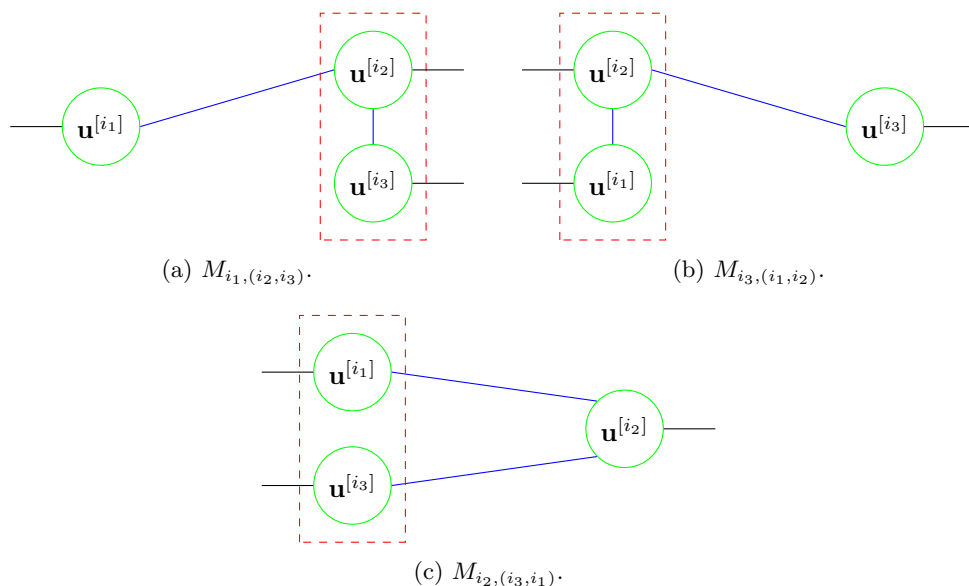
(a) $M_{i_1,(i_2,i_3)}$.

(b) $M_{i_3,(i_1,i_2)}$.

(c) $M_{i_2,(i_3,i_1)}$.

FIG. 6. *Illustration of the intuition of Algorithm 3.3: When (i_1, i_2, i_3) is of the correct order, the rank of $M_{i_1,(i_2,i_3)}$ and $M_{i_3,(i_1,i_2)}$ is at most R, while the rank of $M_{i_2,(i_3,i_1)}$ is generically at least R^2. The reason is that the number of blue lines connecting the red box and the node outside the red box is one in* (a) *and* (b) *but two in* (c), *where red boxes group the column indices in the matricization.*

Algorithm 3.3. Determine the order of three indices for the TT format.

1: **Input:** Three indices $i_1, i_2, i_3 \in \{1, 2, \ldots, d\}$, $R \in \mathbb{N}_+$ approximating the bond dimension, and an oracle computing entries of $\mathbf{T} = \varphi(\vec{\mathbf{u}}, \tau)$.
2: Sample $y_i \in \{1, 2, \ldots, n_i\}$ for $i \in \{1, 2, \ldots, d\} \backslash \{i_1, i_2, i_3\}$.
3: Construct matrices $M_{i_1,(i_2,i_3)}$, $M_{i_2,(i_3,i_1)}$, and $M_{i_3,(i_1,i_2)}$ via (3.4), (3.5), and (3.6).
4: **if** $\sigma_{R^2}(M_{i_1,(i_2,i_3)}) > \max \left\{ \sigma_{R^2}(M_{i_2,(i_3,i_1)}), \sigma_{R^2}(M_{i_3,(i_1,i_2)}) \right\}$ **then**
5:      Return (i_2, i_1, i_3).
6: **end if**
7: **if** $\sigma_{R^2}(M_{i_2,(i_3,i_1)}) > \max \left\{ \sigma_{R^2}(M_{i_3,(i_1,i_2)}), \sigma_{R^2}(M_{i_1,(i_2,i_3)}) \right\}$ **then**
8:      Return (i_1, i_2, i_3).
9: **end if**
10: **if** $\sigma_{R^2}(M_{i_3,(i_1,i_2)}) > \max \left\{ \sigma_{R^2}(M_{i_1,(i_2,i_3)}), \sigma_{R^2}(M_{i_2,(i_3,i_1)}) \right\}$ **then**
11:      Return (i_2, i_3, i_1).
12: **end if**

---

4. Analysis of the algorithms. In this section, we establish the theoretical guarantees for the algorithms displayed in section 3. In section 4.1, we analyze the noiseless case and establish that Algorithms 3.1 3.2, 3.3, and 3.4 recover the correct order unless the data lie in a measure-zero set with respect to the Lebesgue measure (note that $\mathcal{U}_{\vec{r},\vec{n},\tau}^d$ is a Euclidean space). Section 4.2 considers the noisy case and provides high-probability guarantees when the observation is contaminated by Gaussian noise.

4.1. Recovery in the noiseless case. We show that Algorithm 3.1 returns the correct order of any four given indices when the observation is noiseless (Theorem 4.2).

Algorithm 3.4. Recover the underlying graph for the TT format.

1: **Input:** $R \in \mathbb{N}_+$ approximating the bond dimension and a oracle computing entries of $\mathbf{T} = \varphi(\vec{\mathbf{u}}, \tau)$.
2: Let (i_1, i_2, i_3) be the output of Algorithm 3.3 with the input three indices being $1, 2, 3$.
3: **for** $t = 3 : d - 1$ **do**
4:     $j_{\min} = 0$, $j_{\max} = t + 1$. {Use the idea of binary search to find a location in $(i_1, i_2, \ldots, i_t)$ such that inserting $t + 1$ maintains the consistency with τ.}
5:     **while** $j_{\max} - j_{\min} \geq 2$ **do**
6:       **if** $j_{\max} - j_{\min} \geq 3$ **then**
7:         $j_1 = j_{\min} + \lfloor \frac{1}{3} \cdot (j_{\max} - j_{\min}) \rfloor$, $j_2 = j_{\min} + \lfloor \frac{2}{3} \cdot (j_{\max} - j_{\min}) \rfloor$.
8:       **else if** $j_{\min} \geq 1$ **then**
9:         $j_1 = j_{\min}$, $j_2 = j_{\min} + 1$.
10:       **else**
11:         $j_1 = 1$, $j_2 = 2$.
12:       **end if**
13:       Run Algorithm 3.3 with input being $t + 1$, i_{j_1}, and i_{j_2}.
14:       **if** the output of Algorithm 3.3 is $(t + 1, i_{j_1}, i_{j_2})$ or $(i_{j_2}, i_{j_1}, t + 1)$ **then**
15:         $j_{\max} = j_1$.
16:       **else if** the output of Algorithm 3.3 is $(i_{j_1}, t + 1, i_{j_2})$ or $(i_{j_2}, t + 1, i_{j_1})$ **then**
17:         $j_{\min} = j_1$, $j_{\max} = j_2$.
18:       **else**
19:         $j_{\min} = j_2$.
20:       **end if**
21:     **end while**
22:     $(i_1, i_2, \ldots, i_{t+1}) = (i_1, i_2, \ldots, i_{j_{\min}}, t + 1, i_{j_{\max}}, i_{j_{\max}+1}, \ldots, i_t)$.
23: **end for**
24: Return $\tau' \in S_d$ with $\tau'(j) = i_j$, $1 \leq j \leq d$.

This result further implies that Algorithm 3.2 returns the correct complete order of all d nodes (Corollary 4.3).

Our analysis requires the assumption that the bond dimension $\vec{r}$ is relatively small compared to the physical dimension $\vec{n}$. Although this does not cover all tensors in TT/TR format, there exist some interesting tensors with small bond dimensions, such as the TR representing the Ising spin glass in [17] and the discrete Laplace with TT format in [27]. In addition, we require that the minimal element in the bond dimension $\vec{r}$ is comparable with the maximal one. More precise statements of the assumptions can be found in Assumption 4.1 for the TR format (and in Assumption 4.7 for the TT format).

Assumption 4.1. The input $R \in \mathbb{N}_+$ in Algorithms 3.1 and 3.2 satisfies

$$\min_{1 \leq i \leq d} n_i \geq R^2 > \max_{1 \leq j \leq d} r_j \quad \text{and} \quad \min_{1 \leq j \leq d} r_j \geq R.$$

THEOREM 4.2. *For the TR format, suppose that Assumption* 4.1 *holds and one can compute the entries of* $\mathbf{T} = \varphi(\vec{\mathbf{u}}, \tau)$ *exactly. Then, for any* $\tau \in S_d$ *and any four indices* $i_1, i_2, i_3, i_4 \in \{1, 2, \ldots, d\}$, *the output of Algorithm* 3.1 *is of the correct order with respect to* τ *unless* $\vec{\mathbf{u}}$ *is in a measure-zero set of* $\mathcal{U}_{\vec{r}, \vec{n}, \tau}^d$.

COROLLARY 4.3. *In the same setting as in Theorem 4.2, for any $\tau \in S_d$, Algorithm 3.2 can return some element in $\mathcal{CS}_{TR}^d(\tau)$ unless $\vec{\mathbf{u}}$ is in a measure-zero set of $\mathcal{U}_{\vec{r},\vec{n},\tau}^d$.*

For proving Theorem 4.2, we assume that (i_1, i_2, i_3, i_4) is of the correct order with respect to τ, $i_s = \tau(j_s)$ for $s = 1, 2, 3, 4$ and $1 \leq j_1 < j_2 < j_3 < j_4 \leq d$ without loss of generality. The following Lemmas 4.4 and 4.5 establish the upper bounds of $\operatorname{rank}(M_{(i_1, i_2),(i_3, i_4)})$ and $\operatorname{rank}(M_{(i_1, i_4),(i_2, i_3)})$ and a lower bound of $\operatorname{rank}(M_{(i_1, i_3),(i_2, i_4)})$, respectively.

LEMMA 4.4. *Suppose that $1 \leq j_1 < j_2 < j_3 < j_4 \leq d$ and that $i_s = \tau(j_s)$, $s = 1, 2, 3, 4$. Then, for any $\vec{\mathbf{u}} \in \mathcal{U}_{\vec{r},\vec{n},\tau}^d$, it holds that*

$$(4.1) \qquad \operatorname{rank}(M_{(i_1, i_2),(i_3, i_4)}) \leq \min_{j_2+1 \leq j \leq j_3} r_j \cdot \min_{j_4+1 \leq j \leq j_1} r_j$$

and similarly that

$$(4.2) \qquad \operatorname{rank}(M_{(i_1, i_4),(i_2, i_3)}) \leq \min_{j_1+1 \leq j \leq j_2} r_j \cdot \min_{j_3+1 \leq j \leq j_4} r_j,$$

where the index j is understood up to mod d.

Proof. We only prove (4.1). The proof of (4.2) follows the same argument and is thus omitted. Set

$$j' = \arg \min_{j_2+1 \leq j \leq j_3} r_j \quad \text{and} \quad j'' = \arg \min_{j_4+1 \leq j \leq j_1} r_j.$$

The main idea of the proof is to express the matricization of $\mathbf{T} = \varphi(\vec{\mathbf{u}}, \tau)$ with two indices grouping $x_{\tau(j'')}, x_{\tau(j''+1)}, \ldots, x_{\tau(j'-1)}$ and $x_{\tau(j')}, x_{\tau(j'+1)}, \ldots, x_{\tau(j''-1)}$ as the product of two matrices through reindexing of the TR format. The sizes of two matrices are $(n_{\tau(j'')} n_{\tau(j''+1)} \cdots n_{\tau(j'-1)}) \times (r_{j'} r_{j''})$ and $(r_{j'} r_{j''}) \times (n_{\tau(j')} n_{\tau(j'+1)} \cdots n_{\tau(j''-1)})$, respectively. More specifically, consider

$$\tilde{M} \in \mathbb{R}^{(n_{\tau(j'')} n_{\tau(j''+1)} \cdots n_{\tau(j'-1)}) \times (n_{\tau(j')} n_{\tau(j'+1)} \cdots n_{\tau(j''-1)})}$$

defined via

$$\begin{aligned}
\tilde{M}\big((x_{\tau(j'')}, x_{\tau(j''+1)}, &\ldots, x_{\tau(j'-1)}), (x_{\tau(j')}, x_{\tau(j'+1)}, \ldots, x_{\tau(j''-1)}) \big) \\
&= \tau(\vec{\mathbf{u}}, \tau)(x_1, x_2, \ldots, x_d) \\
&= \operatorname{tr}(\mathbf{u}^{[\tau(1)]}(x_{\tau(1)}) \mathbf{u}^{[\tau(2)]}(x_{\tau(2)}) \cdots \mathbf{u}^{[\tau(d)]}(x_{\tau(d)})) \\
&= \operatorname{tr}\Big(\mathbf{u}^{[\tau(j'')]}(x_{\tau(j'')}) \mathbf{u}^{[\tau(j''+1)]}(x_{\tau(j''+1)}) \cdots \mathbf{u}^{[\tau(j'-1)]}(x_{\tau(j'-1)}) \\
&\qquad \cdot \mathbf{u}^{[\tau(j')]}(x_{\tau(j')}) \mathbf{u}^{[\tau(j'+1)]}(x_{\tau(j'+1)}) \cdots \mathbf{u}^{[\tau(j''-1)]}(x_{\tau(j''-1)}) \Big)
\end{aligned}$$

for $1 \leq x_i \leq n_i$, $1 \leq i \leq d$. Note that

$$\mathbf{u}^{[\tau(j'')]}(x_{\tau(j'')}) \mathbf{u}^{[\tau(j''+1)]}(x_{\tau(j''+1)}) \cdots \mathbf{u}^{[\tau(j'-1)]}(x_{\tau(j'-1)}) \in \mathbb{R}^{r_{j''} \times r_{j'}}$$

and

$$\mathbf{u}^{[\tau(j')]}(x_{\tau(j')}) \mathbf{u}^{[\tau(j'+1)]}(x_{\tau(j'+1)}) \cdots \mathbf{u}^{[\tau(j''-1)]}(x_{\tau(j''-1)}) \in \mathbb{R}^{r_{j'} \times r_{j''}}.$$

Let us define two matrices

$$N_1 \in \mathbb{R}^{(n_{\tau(j'')} n_{\tau(j''+1)} \cdots n_{\tau(j'-1)}) \times (r_{j'} r_{j''})}$$

and

$$N_2 \in \mathbb{R}^{(r_{j'} r_{j''}) \times (n_{\tau(j')} n_{\tau(j'+1)} \cdots n_{\tau(j''-1)})}$$

as

$$
\begin{aligned}
&N_1\big((x_{\tau(j'')}, x_{\tau(j''+1)}, \ldots, x_{\tau(j'-1)}), (k_{j'}, k_{j''})\big) \\
&\quad := \Big(\mathbf{u}^{[\tau(j'')]}(x_{\tau(j'')}) \mathbf{u}^{[\tau(j''+1)]}(x_{\tau(j''+1)}) \cdots \mathbf{u}^{[\tau(j'-1)]}(x_{\tau(j'-1)})\Big)_{k_{j''}, k_{j'}}
\end{aligned}
$$

and

$$
\begin{aligned}
&N_2\big((k_{j'}, k_{j''}), (x_{\tau(j')}, x_{\tau(j'+1)}, \ldots, x_{\tau(j''-1)})\big) \\
&\quad := \Big(\mathbf{u}^{[\tau(j')]}(x_{\tau(j')}) \mathbf{u}^{[\tau(j'+1)]}(x_{\tau(j'+1)}) \cdots \mathbf{u}^{[\tau(j''-1)]}(x_{\tau(j''-1)})\Big)_{k_{j'}, k_{j''}},
\end{aligned}
$$

where $1 \le x_i \le n_i$, $1 \le i \le d$, and $1 \le k_{j'} \le r_{j'}$, $1 \le k_{j''} \le r_{j''}$. Therefore, it holds that

$$
\begin{aligned}
&\tilde{M}\big((x_{\tau(j'')}, x_{\tau(j''+1)}, \ldots, x_{\tau(j'-1)}), (x_{\tau(j')}, x_{\tau(j'+1)}, \ldots, x_{\tau(j''-1)})\big) \\
&= \operatorname{tr}\Big(\mathbf{u}^{[\tau(j'')]}(x_{\tau(j'')}) \mathbf{u}^{[\tau(j''+1)]}(x_{\tau(j''+1)}) \cdots \mathbf{u}^{[\tau(j'-1)]}(x_{\tau(j'-1)}) \\
&\qquad\quad \cdot \mathbf{u}^{[\tau(j')]}(x_{\tau(j')}) \mathbf{u}^{[\tau(j'+1)]}(x_{\tau(j'+1)}) \cdots \mathbf{u}^{[\tau(j''-1)]}(x_{\tau(j''-1)})\Big) \\
&= \sum_{k_{j'}=1}^{r_{j'}} \sum_{k_{j''}=1}^{r_{j''}} \Big(\mathbf{u}^{[\tau(j'')]}(x_{\tau(j'')}) \mathbf{u}^{[\tau(j''+1)]}(x_{\tau(j''+1)}) \cdots \mathbf{u}^{[\tau(j'-1)]}(x_{\tau(j'-1)})\Big)_{k_{j''}, k_{j'}} \\
&\qquad\qquad \cdot \Big(\mathbf{u}^{[\tau(j')]}(x_{\tau(j')}) \mathbf{u}^{[\tau(j'+1)]}(x_{\tau(j'+1)}) \cdots \mathbf{u}^{[\tau(j''-1)]}(x_{\tau(j''-1)})\Big)_{k_{j'}, k_{j''}} \\
&= N_1\big((x_{\tau(j'')}, x_{\tau(j''+1)}, \ldots, x_{\tau(j'-1)}), :\big) N_2\big(:, (x_{\tau(j')}, x_{\tau(j'+1)}, \ldots, x_{\tau(j''-1)})\big).
\end{aligned}
$$

This implies that $\tilde{M} = N_1 N_2$ and hence that $\operatorname{rank}(\tilde{M}) \le r_{j'} r_{j''}$, which implies (4.1) because $M_{(i_1, i_2), (i_3, i_4)}$ is a submatrix of $\tilde{M}$. □

LEMMA 4.5. *Suppose that $1 \le j_1 < j_2 < j_3 < j_4 \le d$ and that $i_s = \tau(j_s)$, $s = 1, 2, 3, 4$. Let R_1, R_2, R_3, R_4 satisfy*

$$R_s \le \min_{j_s+1 \le j \le j_{s+1}} r_j \quad and \quad n_{i_s} \ge R_{s-1} R_s, \quad 1 \le s \le 4,$$

where the subscript of $R_\cdot$ and $j_\cdot$ is understood via mod 4. Then, the following holds unless $\vec{u}$ is in a measure-zero set of $\mathcal{U}_{\vec{r}, \vec{n}, \tau}^d$:

$$(4.3) \qquad\qquad \operatorname{rank}(M_{(i_1, i_3), (i_2, i_4)}) \ge R_1 R_2 R_3 R_4.$$

Remark 4.6. The main technique used in the proof of Lemma 4.5 is to establish the equivalence between the points in $\mathcal{U}_{\vec{r}, \vec{n}, \tau}^d$ violating (4.3) with common roots of a class of polynomials. Note that the measure of the root set of a nonzero polynomial is zero; one only needs to show that some polynomial is nonzero, for which constructing a point that is not a common root suffices.

Proof of Lemma 4.5. One has $n_{i_1} n_{i_3} \ge \tilde{R}$ and $n_{i_2} n_{i_4} \ge \tilde{R}$ with $\tilde{R} = R_1 R_2 R_3 R_4$. We denote $N = \binom{n_1 n_2}{\tilde{R}} \cdot \binom{n_3 n_4}{\tilde{R}}$, which is the number of $R \times R$ minors in a $n_1 n_2 \times n_3 n_4$ matrix. Consider

$$
\begin{aligned}
f_\ell : \mathcal{U}_{\vec{r}, \vec{n}, \tau}^d &\to \mathbb{R}, \\
\vec{u} &\mapsto \det(M_{(i_1, i_3), (i_2, i_4)}^{(\ell)}),
\end{aligned}
$$

where $M_{(i_1,i_3),(i_2,i_4)}^{(\ell)}$ denotes the ℓth $R \times R$ minor of $M_{(i_1,i_3),(i_2,i_4)}$, for $1 \le \ell \le N$. Observe that f_ℓ is a homogeneous polynomial over entries of $\vec{\mathbf{u}}$.

For $\vec{\mathbf{u}} \in \mathcal{U}_{\vec{r},\vec{n},\tau}^d$, the corresponding $M_{(i_1,i_3),(i_2,i_4)}$ satisfies $\operatorname{rank}(M_{(i_1,i_3),(i_2,i_4)}) < R$ if and only if

$$\vec{\mathbf{u}} \in \mathbb{V}(f_1, f_2, \ldots, f_N) \subset \mathcal{U}_{\vec{r},\vec{n},\tau}^d,$$

where $\mathbb{V}(f_1, f_2, \ldots, f_N)$ is the zero locus of the ideal generated by $\{f_1, f_2, \ldots, f_N\}$, i.e., the set of common roots of $f_1, f_2, \ldots, f_N$. To prove that $\mathbb{V}(f_1, f_2, \ldots, f_N)$ is of measure zero, it suffices to show that at least one of $f_1, f_2, \ldots, f_N$ is a nonzero polynomial. Therefore, one only needs to construct a $\vec{\mathbf{u}} \in \mathcal{U}_{\vec{r},\vec{n},\tau}^d$ satisfying $\vec{\mathbf{u}} \notin \mathbb{V}(f_1, f_2, \ldots, f_N)$; i.e., the rank of the matrix $M_{(i_1,i_3),(i_2,i_4)}$ associated with $\vec{\mathbf{u}}$ is at least $\tilde{R}$.

Let us then construct such a $\vec{\mathbf{u}} = (\mathbf{u}^{[1]}, \mathbf{u}^{[2]}, \ldots, \mathbf{u}^{[d]})$. We denote $e_{p,r}$ as the vector in $\mathbb{R}^r$ with the pth entry being 1 and other entries being 0. Set

$$
\begin{aligned}
&\mathbf{u}^{[\tau(j_s)]}(x_{\tau(j_s)}) \\
&= \begin{cases} e_{p,r_{j_s}} e_{q,r_{j_s+1}}^\top & \text{if } x_{\tau(j_s)} = (p-1)R_s + q, \ 1 \le p \le R_{s-1}, \ 1 \le q \le R_s, \\ 0_{r_{j_s} \times r_{j_s+1}} & \text{if } R_{s-1}R_s < x_{\tau(j_s)} \le n_{i_s} \end{cases}
\end{aligned}
$$

for $s = 1, 2, 3, 4$, and

$$\mathbf{u}^{[i]}(y_i) = \sum_{p=1}^{R_s} e_{p,r_{\tau^{-1}(i)}} e_{p,r_{\tau^{-1}(i)+1}}^\top$$

for $i \in \{1, 2, \ldots, d\} \backslash \{i_1, i_2, i_3, i_4\}$, where s is the unique index in $\{1,2,3,4\}$ such that $j_s < \tau^{-1}(i) < j_{s+1}$. Then, we can compute for $x_{\tau(j_s)} = (p_s - 1)R_s + q_s$, where $1 \le p_s \le R_{s-1}$, $1 \le q_s \le R_s$, and $1 \le s \le 4$, that

$$
\begin{aligned}
&M_{(i_1,i_3),(i_2,i_4)}((x_{i_1}, x_{i_3}), (x_{i_3 2}, x_{i_4})) \\
&= M_{(i_1,i_3),(i_2,i_4)}((x_{\tau(j_1)}, x_{\tau(j_3)}), (x_{\tau(j_2)}, x_{\tau(j_4)})) \\
&= \operatorname{tr}\left(\prod_{s-1}^{4} \left(\mathbf{u}^{[\tau(j_s)]}(x_{\tau(j_s)}) \mathbf{u}^{[\tau(j_s+1)]}(y_{\tau(j_s+1)}) \cdots \mathbf{u}^{[\tau(j_{s+1}-1)]}(y_{\tau(j_{s+1}-1)}) \right) \right) \\
&= \operatorname{tr}\left(\prod_{s-1}^{4} \left(e_{p_s,r_{j_s}} e_{q_s,r_{j_s+1}}^\top \prod_{j=j_s+1}^{j_{s+1}-1} \sum_{p=1}^{R_s} e_{p,r_j} e_{p,r_{j+1}}^\top \right) \right) \\
&= \operatorname{tr}\left(e_{p_1,r_{j_1}} e_{q_1,r_{j_1+1}}^\top \left(\sum_{p=1}^{R_1} e_{p,r_{j_1+1}} e_{p,r_{j_2}}^\top \right) e_{p_2,r_{j_2}} e_{q_2,r_{j_2+1}}^\top \left(\sum_{p=1}^{R_2} e_{p,r_{j_2+1}} e_{p,r_{j_3}}^\top \right) \right. \\
&\qquad \left. \cdot e_{p_3,r_{j_3}} e_{q_3,r_{j_3+1}}^\top \left(\sum_{p=1}^{R_3} e_{p,r_{j_3+1}} e_{p,r_{j_4}}^\top \right) e_{p_4,r_{j_4}} e_{q_4,r_{j_4+1}}^\top \left(\sum_{p=1}^{R_4} e_{p,r_{j_4+1}} e_{p,r_{j_1}}^\top \right) \right) \\
&= \prod_{s=1}^{4} e_{q_s,r_{j_s+1}}^\top \left(\sum_{p=1}^{R_s} e_{p,r_{j_s+1}} e_{p,r_{j_s+1}}^\top \right) e_{p_{s+1},r_{j_s+1}} \\
&= \delta_{q_1,p_2} \delta_{q_2,p_3} \delta_{q_3,p_4} \delta_{q_4,p_1},
\end{aligned}
$$

where $\delta_{\cdot,\cdot}$ is the Kronecker delta function: $\delta_{a,b} = 1$ if $a = b$; $\delta_{a,b} = 0$ if $a \ne b$. Note that

$$M_{(i_1,i_3),(i_2,i_4)}((x_{i_1}, x_{i_3}), (x_{i_2}, x_{i_4})) = M_{(i_1,i_3),(i_2,i_4)}((x_{\tau(j_1)}, x_{\tau(j_3)}), (x_{\tau(j_2)}, x_{\tau(j_4)})) = 0$$

if there exists $s \in \{1, 2, 3, 4\}$ such that $x_{\tau(j_s)} > R_{s-1} R_s$. Therefore, we obtain that

$$\operatorname{rank}(M_{(i_1, i_3), (i_2, i_4)}) = R_1 R_2 R_3 R_4 = \tilde{R},$$

which completes the proof. □

Proof of Theorem 4.2. In Lemma 4.4 and Lemma 4.5, we have shown that, if Assumption 4.1 holds and (i_1, i_2, i_3, i_4) is of the correct order with respect to τ, then almost surely,

$$(4.4) \quad \operatorname{rank}(M_{(i_1, i_3), (i_2, i_4)}) \geq R^4 > \max \left\{ \operatorname{rank}(M_{(i_1, i_2), (i_3, i_4)}), \operatorname{rank}(M_{(i_1, i_4), (i_2, i_3)}) \right\}.$$

This guarantees the correctness of Algorithm 3.1. □

The correctness of Algorithms 3.3 and 3.4 can be established with similar proof techniques. We state the result in Theorem 4.8 and Corollary 4.9 and defer the proof of Theorem 4.8 to Appendix B.

Assumption 4.7. There exists some $R \in \mathbb{N}_+$ such that

$$\min\{n_{\tau(1)}, n_{\tau(d)}\} \geq R, \quad \min_{2 \leq j \leq d-1} n_{\tau(j)} \geq R^2 > \max_{2 \leq j \leq d} r_j \geq \min_{2 \leq j \leq d} r_j \geq R.$$

THEOREM 4.8. *For the TT format, suppose that Assumption* 4.7 *holds and one can compute the entries of* $\mathbf{T} = \varphi(\vec{\mathbf{u}}, \tau)$ *exactly. Then, for any* $\tau \in S_d$ *and any three indices* $i_1, i_2, i_3 \in \{1, 2, \ldots, d\}$, *the output of Algorithm* 3.3 *is of the correct order with respect to* τ *unless* $\vec{\mathbf{u}}$ *is in a measure-zero set of* $\mathcal{U}_{\vec{r}, \vec{n}, \tau}^d$.

COROLLARY 4.9. *In the same setting as in Theorem* 4.8, *for any* $\tau \in S_d$, *Algorithm* 3.4 *can return some element in* $\mathcal{CS}_{TT}^d(\tau)$ *unless* $\vec{\mathbf{u}}$ *is in a measure-zero set of* $\mathcal{U}_{\vec{r}, \vec{n}, \tau}^d$.

4.2. Robustness against observation error. Because Algorithms, 3.1, 3.2, 3.3, and 3.4 are based on singular values, which are continuous with respect to matrix entries, they are expected to be robust against observational noise. In this section, we establish rigorous robustness results assuming the noise is Gaussian. Similar to the previous subsection, we will present the proof for the TR format and defer the similar proof for the TT format to Appendix A.

Let $\tau \in S_d$ be the underlying permutation, and let R be as in Assumption 4.1. It has already been proved in Lemma 4.5 that

$$(4.5) \qquad\qquad \sigma_{R^4}(M_{(i_1, i_3), (i_2, i_4)}) > 0$$

for almost all $\vec{\mathbf{u}} \in \mathcal{U}_{\vec{r}, \vec{n}, \tau}^d$ and all $i_1, i_2, i_3, i_4 \in \{1, 2, \ldots, d\}$ of the correct order with respect to τ. For the noisy case, we need a stronger assumption than (4.5), requiring a uniform positive lower bound of the R^4th singular value.

Assumption 4.10. Let $\vec{\mathbf{u}} \in \mathcal{U}_{\vec{r}, \vec{n}, \tau}^d$ be the underlying data for the TR format, and let Assumption 4.1 be satisfied with $R > 0$. There exists a positive constant $\sigma > 0$ such that, for any $i_1, i_2, i_3, i_4 \in \{1, 2, \ldots, d\}$ of correct order with respect to τ and any $y_i \in \{1, 2, \ldots, n_i\}$, $i \in \{1, 2, \ldots, d\} \backslash \{i_1, i_2, i_3, i_4\}$, the matrix $M_{(i_1, i_3), (i_2, i_4)} \in \mathbb{R}^{(n_{i_1} n_{i_3}) \times (n_{i_2} n_{i_4})}$ satisfies

$$(4.6) \qquad\qquad \sigma_{R^4}(M_{(i_1, i_3), (i_2, i_4)}) \geq \sigma.$$

With the uniform bound assumption above, we can establish a high-probability guarantee of the correctness of Algorithm 3.1.

THEOREM 4.11. *Suppose that Assumption* 4.10 *holds and the noise in each observation of* $\mathbf{T} = \varphi(\vec{\mathbf{u}}, \tau)$ *satisfies* $\mathcal{N}(0, \sigma_e^2)$ *independently. For any* $T < \frac{\sigma}{\sigma_e}$ *(σ is defined in* (4.6)*), any four indices* $i_1, i_2, i_3, i_4 \in \{1, 2, \ldots, d\}$, *and any* $y_i \in \{1, 2, \ldots, n_i\}$, $i \in \{1, 2, \ldots, d\} \setminus \{i_1, i_2, i_3, i_4\}$, *the probability that Algorithm* 3.1 *returns an incorrect ordering of* i_1, i_2, i_3, i_4 *is no more than*

$$6 \cdot \exp\left(-\frac{1}{8} \max\{T - 4n_{\max}, 0\}^2\right).$$

COROLLARY 4.12. *In the same setting as in Theorem* 4.11*, the probability that Algorithm* 3.2 *returns an incorrect ordering is smaller than or equal to*

$$Cd \log d \cdot \exp\left(-\frac{1}{8} \max\{T - 4n_{\max}, 0\}^2\right),$$

where C *is an absolute constant.*

It is evident from Theorem 4.11 and Corollary 4.12 that the level of robustness against the noise is largely determined by the singular value threshold σ in (4.6). In some datasets, σ may be small, a scenario to be demonstrated through numerical examples later.

For proving Theorem 4.11, we first state some preliminaries that bound the singular values of a matrix with noisy entries.

THEOREM 4.13 (largest singular value of matrices with independent and identically distributed (i.i.d.) Gaussian entries, [29, 9]). *Suppose that* $A \in \mathbb{R}^{m_1 \times m_2}$ *and that the entries of* A *are i.i.d.* $\mathcal{N}(0, 1)$. *Then, the largest singular value of* A *satisfies*

$$\mathbb{P}(\sigma_{\max}(A) > \sqrt{m_1} + \sqrt{m_2} + t) \leq 2e^{-t^2/2} \quad \forall\, t > 0.$$

THEOREM 4.14 (Weyl's inequality). *For any* $A, E \in \mathbb{R}^{m_1 \times m_2}$, *it holds that*

$$|\sigma_k(A + E) - \sigma_k(A)| \leq \sigma_{\max}(E) \quad \forall\, 1 \leq k \leq \min\{m_1, m_2\}.$$

Proof of Theorem 4.11. We first show the robustness of Algorithm 3.1. Without loss of generality, let us assume that (i_1, i_2, i_3, i_4) is of the correct order with respect to τ. Let E be the noise matrix in observing $M_{(i_1, i_2),(i_3, i_4)}$, $M_{(i_1, i_3),(i_2, i_4)}$, or $M_{(i_1, i_4),(i_2, i_3)}$. Then, by Theorem 4.13, it holds that

$$\mathbb{P}\left(\sigma_{\max}(E) \geq \frac{\sigma}{2}\right) \leq \mathbb{P}\left(\sigma_{\max}(E) > (2n_{\max} + t)\sigma_e\right) \leq 2e^{-t^2/2},$$

where $t = \frac{T}{2} - 2n_{\max}$. Then, using Assumption 4.10 and Theorem 4.14 (Weyl's inequality) with probability at least

$$1 - 6 \cdot \exp\left(-\frac{t^2}{2}\right) = 1 - 6 \cdot \exp\left(-\frac{1}{8}(T - 4n_{\max})^2\right),$$

we have $\sigma_{R^4}(M_{(i_1, i_2),(i_3, i_4)}) < \frac{\sigma}{2}$, $\sigma_{R^4}(M_{(i_1, i_3),(i_2, i_4)}) > \frac{\sigma}{2}$, and $\sigma_{R^4}(M_{(i_1, i_4),(i_2, i_3)}) < \frac{\sigma}{2}$, which leads to the correctness of the output of Algorithm 3.1. $\square$

Proof of Corollary 4.12. Corollary 4.12 follows immediately from Theorem 4.11 and the fact that Algorithm 3.2 calls Algorithm 3.1 for $O(d \log d)$ times. $\square$

The robustness result of Algorithm 3.3 and Algorithm 3.4 is similar, which is stated as follows and proved in Appendix B.

Assumption 4.15. Let $\vec{\mathbf{u}} \in \mathcal{U}_{\vec{r}, \vec{n}, \tau}^d$ be the underlying data for the TT format, and let Assumption 4.7 be satisfied with $R > 0$. There exists a positive constant $\sigma > 0$ such that, for any $i_1, i_2, i_3 \in \{1, 2, \ldots, d\}$ of correct order with respect to τ and any

$y_i \in \{1, 2, \ldots, n_i\}$, $i \in \{1, 2, \ldots, d\} \backslash \{i_1, i_2, i_3\}$, the matrix $M_{i_2,(i_1,i_3)} \in \mathbb{R}^{n_{i_2} \times (n_{i_1} n_{i_3})}$ satisfies

$$(4.7) \qquad\qquad \sigma_{R^2}(M_{i_2,(i_1,i_3)}) \geq \sigma.$$

THEOREM 4.16. *Suppose that Assumption* 4.15 *holds and the noise in each observation of* $\mathbf{T} = \varphi(\vec{\mathbf{u}}, \tau)$ *satisfies* $\mathcal{N}(0, \sigma_e^2)$ *independently. For any* $T < \frac{\sigma}{\sigma_e}$ *(σ is defined in* (4.7)*), any three indices* $i_1, i_2, i_3 \in \{1, 2, \ldots, d\}$, *and any* $y_i \in \{1, 2, \ldots, n_i\}$, $i \in \{1, 2, \ldots, d\} \backslash \{i_1, i_2, i_3\}$, *the probability that Algorithm* 3.3 *returns an incorrect ordering of* i_1, i_2, i_3 *is at most*

$$6 \cdot \exp\left(-\frac{1}{8} \max\{T - 2n_{\max} - 2\sqrt{n_{\max}}, 0\}^2 \right)$$

as long as the noise in each observation of $\mathbf{T} = \varphi(\vec{\mathbf{u}}, \tau)$ *distributes as* $\mathcal{N}(0, \sigma_e^2)$ *independently with* $\sigma_e < \frac{\sigma}{T}$, *where* σ *is the lower bound in* (4.7).

COROLLARY 4.17. *In the same setting as in Theorem* 4.16, *the probability that Algorithm* 3.4 *returns an incorrect ordering is smaller than or equal to*

$$C d \log d \cdot \exp\left(-\frac{1}{8} \max\{T - 2n_{\max} - 2\sqrt{n_{\max}}, 0\}^2 \right),$$

where C *is an absolute constant.*

5. Numerical experiments. We present some numerical results in this section. Section 5.1 is for implementing the proposed tensor order recovery algorithms, which shows the correctness and efficiency of the proposed approach. In section 5.2, we compare our work with [1], which heuristically identifies the structure of tree tensor networks. We evaluate our TR order recovery algorithm on a more practical model, the Potts model, in section 5.3.

5.1. Implementation of the proposed algorithms for TR/TT formats. Throughout this subsection, we set $d = 8$ and $\vec{n} = (4, 4, \ldots, 4)$. $\vec{r}$ is chosen as $(3, 3, \ldots, 3)$ for the TR format and $(1, 3, 3, \ldots, 3)$ for the TT format. One can verify that Assumption 4.1 and Assumption 4.7 are satisfied for $R = 2$. For a fixed noise level σ_e, we repeat the following procedure for 10000 times:
- Sample an underlying permutation from the uniform distribution $\tau \sim \mathcal{U}(S_d)$.
- Sample an element $\vec{\mathbf{u}} \in \mathcal{U}_{\vec{r}, \vec{n}, \tau}^d$ with each entry of $\vec{\mathbf{u}}$ being i.i.d. $\mathcal{N}(0, 1)$.
- Run Algorithm 3.2, Algorithm 3.4, or their majority-vote versions with 5 voters, where each entry of $\mathbf{T} = \varphi(\vec{\mathbf{u}}, \tau)$ is observed with noise from $\mathcal{N}(0, \sigma_e^2)$. Denote the output by τ'.

We count the number of trials with $\tau' \in \mathcal{CS}_{\text{TR}}^d(\tau)$ or $\tau' \in \mathcal{CS}_{\text{TT}}^d(\tau)$ and divide it by 10000 to obtain an estimate of the probability that Algorithm 3.2, Algorithm 3.4, or their majority-vote versions obtain the correct order/underlying graph. We then repeat the experiments with a modified method for generating the data while keeping all other settings unchanged. Specifically, the entries of $\vec{\mathbf{u}}$ are independent with $\mathbf{u}^{[i]}(k_i, x_{\tau(i)}, k_{i+1}) \sim \mathcal{N}(0, 0.1^2)$ if $k_i = 3$ or $k_{i+1} = 3$ and $\mathbf{u}^{[i]}(k_i, x_{\tau(i)}, k_{i+1}) \sim \mathcal{N}(0, 1)$, otherwise. This dataset is nearly rank deficient because it can be seen as a perturbation from a TR format with $\vec{r} = (2, 2, \ldots, 2)$ or a TT format $\vec{r} = (1, 2, \ldots, 2)$, in contrast to the previous dataset, which is viewed full-rank.

The results are shown in Figure 7 for the TR format and in Figure 8 for the TT format, respectively. One can see that the probability of correct recovery is 1 when $\sigma_e = 0$ and is high when σ_e is small, which fits the theory established in section 4.
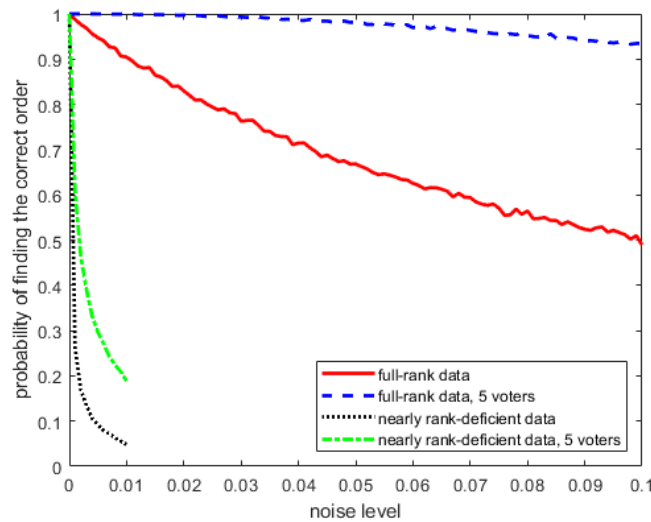
FIG. 7. *Numerical results for TR format. The probability of correct recovery is* 1 *when there is no noise. The robustness is much better on full-rank data compared to nearly rank-deficient data. Using majority vote with* 5 *voters significantly improves the robustness of the algorithm.*
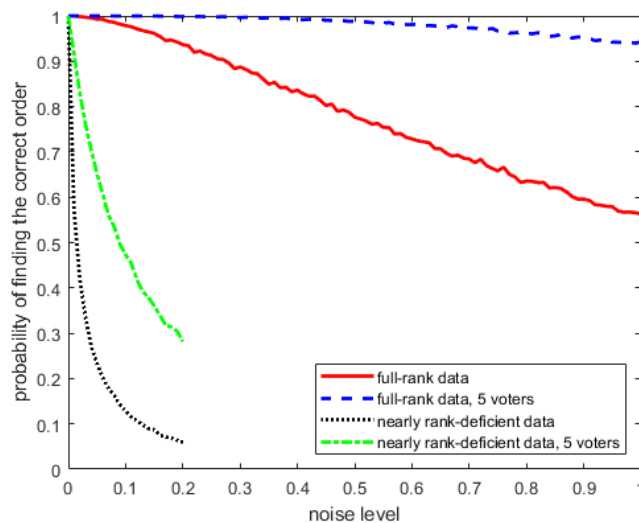

FIG. 8. *Numerical results for TT format. The probability of correct recovery is* 1 *when there is no noise. The proposed method is more robust on full-rank data than the nearly rank-deficient data. Using majority vote with* 5 *voters significantly improves the robustness of the algorithm.*

It is also evident that the robustness on the full-rank data is significantly better than that on the nearly rank-deficient data. As discussed in section 4.2, this is because the singular value threshold is small for the nearly rank-deficient data. In addition, the majority vote can significantly improve the robustness of our approach even if the number of voters is small.

Another observation is that the algorithms for TT format tensor network recovery is more robust against the observation noise than those for the TR format. More specifically, the curves of probability in Figure 7 and Figure 8 are similar, although the maximal noise levels are significantly different: 0.1 for the TR format and 1 for the TT format. This is because, when recovering the partial information (i.e., the

order of 3 or 4 indices), Algorithm 3.1 samples a matrix with size of $n_{\max}^2 \times n_{\max}^2$, which is much larger than the matrix sampled in Algorithm 3.3, whose size is at most $n_{\max} \times n_{\max}^2$. According to Theorem 4.13 and Theorem 4.14, for a fixed noise level, the perturbation of singular values of a matrix has a larger upper bound when the size of the matrix is larger. The perturbation of singular subspaces follows similar rules (see, e.g., [3, Theorem 3]).

5.2. Comparison with previous work for TT format. We compare our approach with the one proposed in [1] in this subsection. The method introduced in [1] aims to recover a general tree structure of a tensor network. The method starts by initializing with a discrete partition $\{\{1\}, \{2\}, \ldots, \{d\}\}$ and subsequently clusters the partition based on some minimal-rank condition. Note that the TR format does not admit a tree structure and hence we only compare our method with the one in [1] for the TT format. An additional insight is that the graph associated with the TT structure is a very special tree, which means it is unnecessary to implement the algorithm in [1] for the most general setting. Alternatively, we first select an endpoint and the TT chain; subsequently, we extend the selected subchain by adding new nodes, guided by the minimal-rank condition. Note that the subchain extension is in the same spirit as clustering the existing subchain with a new node. The numerical comparison between Algorithm 3.4 (with five voters) and that of [1] is shown in Figure 9 for $d = 6$, $\vec{n} = (4, 4, \ldots, 4)$, $\vec{r} = (1, 3, 3, \ldots, 3)$, with other settings being the same as the full-rank experiments in section 5.1. It can be seen that both methods by us and by [1] can recover the correct order in the noiseless case, but our method performs more robustly against the observation error. It is worth noting that we implement the method in [1] while estimating ranks based on observation from all entries of $\mathbf{T}$. This process entails much higher complexity compared to Algorithm 3.4, which only makes queries to down-sampled tensors. These distinctions show our proposed approach is more robust and computationally efficient compared to the approach in [1].
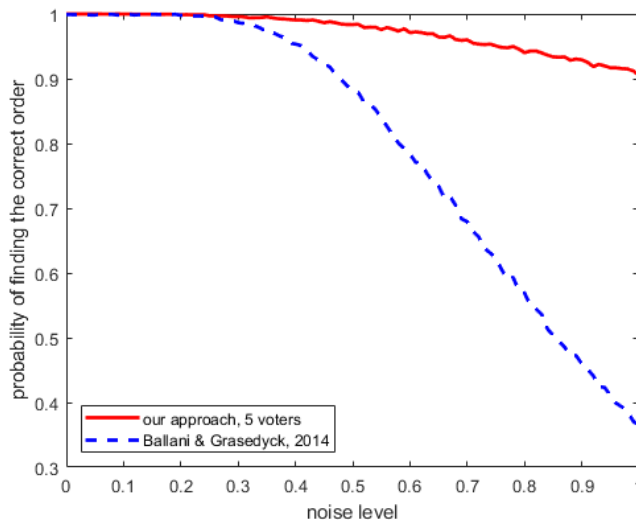

FIG. 9. *Numerical comparison between Algorithm* 3.4 *(with five voters) and the method proposed in* [1] *for tensor train format. We observe all entries of* $\mathbf{T}$ *for the rank estimation when implementing the method in* [1] *and only observe down-sampled tensors when implementing our approach. Even if we allow the method of* [1] *to have larger query complexity, our work still supersedes* [1] *in the sense of having significantly larger probability of finding the correct order.*

5.3. Potts model. We demonstrate the efficacy of our algorithm for identifying the permutation for a Potts model [16, 41] when the underlying geometry is unknown. The Potts model is a generalization of the Ising model used in statistical physics: Consider a many-body system with d sites/spins and r different spin values, and only adjacent sites on a ring (with an unknown permutation $\tau \in S_d$) admit interactions. Specifically, the Hamiltonian is given by

$$H_\tau(k_1, k_2, \ldots, k_d) = \sum_{i=1}^{d} J_{\tau(i)}(k_{\tau(i)}, k_{\tau(i+1)}),$$

where $k_{\tau(i)}$ is the spin value at the ith site on the ring and $J_{\tau(i)} \in \mathbb{R}^{r \times r}$ is a symmetric matrix representing the interaction pattern of the ith and $(i+1)$st sites on the ring. This Hamiltonian leads to the following free energy:

$$(5.1) \qquad f_\tau(J_1, J_2, \ldots, J_d) = -\frac{1}{\beta} \log \left(\sum_{k_1, k_2, \ldots, k_d = 1}^{r} e^{-\beta H_\tau(k_1, k_2, \ldots, k_d)} \right)$$

$$= -\frac{1}{\beta} \log \left(\mathrm{tr} \left(\prod_{i=1}^{d} e^{-\beta J_{\tau(i)}} \right) \right),$$

where β is the inverse temperature and e^A denotes the elementwise exponential of a matrix A.

Our experiment is a generalization of the Ising model experiment in [17]. We set $d = 6$, $r = 3$, $\beta = 10$ and let $J_1, J_2, \ldots, J_d$ take values among $n = 5$ predetermined $r \times r$ symmetric matrices with diagonal and off-diagonal entries sampled i.i.d. from $\mathcal{N}(0, 1)$. The free energy f_τ can thus be viewed as a dth-order tensor with physical dimension being $\vec{n} = (5, \ldots, 5)$, and we assume that the entries of f_τ are observed with i.i.d. noise $\mathcal{N}(0, \sigma_e^2)$ with some noise level σ_e. We then apply Algorithm 3.1 and Algorithm 3.2 with $R = 2$ to recover the underlying permutation τ, and the results of 10000 independent trials are shown in Figure 10.
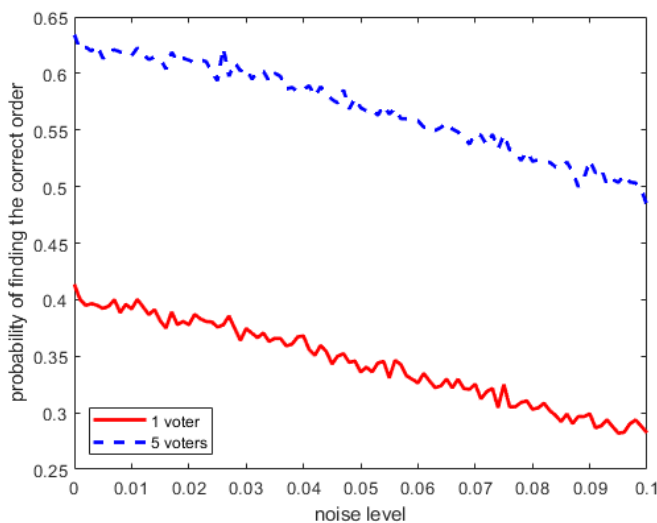

FIG. 10. *Numerical results for the Potts model (3 spin values) with TR format. The probability of correct recovery is relatively high even though the free energy f_τ is not precisely of TR format with low bond dimension.*

It can be seen from Figure 10 that our algorithms can correctly recover the underlying permutation for the Potts model with reasonably high probability. Note that the free energy f_τ in (5.1) is not precisely in the TR format due to the logarithm, which means that Assumption 4.1 does not hold. Consequently, the probability of finding the correct order is not 1 in the noiseless case. Nevertheless, the success probability is still quite high, which further demonstrates the effectiveness of our algorithms.

6. Conclusion and discussions. We propose new algorithms for recovering the underlying graphs of the TR and TT formats by querying entries of the tensor. The proposed algorithms obtain the orders of the selected 3 or 4 indices by constructing matricizations using down-sampling and comparing their ranks. The algorithm follows from the observation that the resulting matricization has a smaller rank if the grouping of indices is compatible with the underlying order of indices. These methods are justified by theory—almost-sure correctness in the noiseless case and high-probability bounds for cases with observation error. In the numerical experiments, we verify these theoretical results and observe that a proper strategy of combining partial information can significantly improve the robustness against the observation error.

We also acknowledge some limitations of this work. First, specific assumptions regarding the bond dimension and the physical dimension are necessary to ensure the provable efficiency of the proposed algorithm and the uniqueness of the underlying permutation. Second, the level of robustness depends on the singular value threshold, which is generally unclear in a quantitative sense. Third, our analysis does not cover incomplete observations, which could be a promising area for future research given existing methods from tensor/matrix completion and SVD for incomplete observations. Another important future direction is the recovery of underlying graphs for general tensor networks. All these topics could be interesting avenues for future research.

Appendix A. Uniqueness of permutations. This section establishes the uniqueness results for the underlying permutation, in the sense that two permutations that can represent the same tensor must be in the same equivalence class defined in (2.5) or (2.6). The results require the same assumptions as in section 4.1 and are stated in Theorem A.1 and Theorem A.2 for the TR and TT formats, respectively.

THEOREM A.1. *Let $\tau, \tau' \in S_d$, and let $\vec{\mathbf{u}} \in \mathcal{U}^d_{\vec{r},\vec{n},\tau}, \vec{\mathbf{u}}' \in \mathcal{U}^d_{\vec{r}',\vec{n},\tau'}$ satisfy $\varphi(\vec{\mathbf{u}},\tau) = \varphi(\vec{\mathbf{u}}',\tau')$. Suppose that $(\vec{r},\vec{n})$ and $(\vec{r}',\vec{n})$ both satisfy Assumption 4.1 with the same $R \in \mathbb{N}_+$. Then, we have $\tau' \in \mathcal{CS}^d_{TR}(\tau)$ unless $\vec{\mathbf{u}}$ or $\vec{\mathbf{u}}'$ is located in some measure-zero subset of $\mathcal{U}^d_{\vec{r},\vec{n},\tau}$ or $\mathcal{U}^d_{\vec{r}',\vec{n},\tau'}$.*

Proof. Suppose that $\tau' \notin \mathcal{CS}^d_{\mathrm{TR}}(\tau)$. Then, there must exist four distinct indices $i_1, i_2, i_3, i_4 \in \{1, 2, \ldots, d\}$ such that (i_1, i_2, i_3, i_4) is of the correct order with respect to τ but is not of the correct order with respect to τ'. By Lemma 4.4 and Lemma 4.5, it holds almost surely that

$$(\text{A.1}) \qquad \operatorname{rank}\left(M'_{(i_1,i_2),(i_3,i_4)}\right) \geq R^4 > \operatorname{rank}\left(M_{(i_1,i_2),(i_3,i_4)}\right),$$

where $M_{(i_1,i_2),(i_3,i_4)}$ and $M'_{(i_1,i_2),(i_3,i_4)}$ are the matricization of $\varphi(\vec{\mathbf{u}},\tau)$ and $\varphi(\vec{\mathbf{u}}',\tau')$ defined in (3.1). However, it follows from $\varphi(\vec{\mathbf{u}},\tau) = \varphi(\vec{\mathbf{u}}',\tau')$ that $M_{(i_1,i_2),(i_3,i_4)} = M'_{(i_1,i_2),(i_3,i_4)}$, which contradicts (A.1). □

THEOREM A.2. *Let $\tau, \tau' \in S_d$, and let $\vec{\mathbf{u}} \in \mathcal{U}^d_{\vec{r},\vec{n},\tau}, \vec{\mathbf{u}}' \in \mathcal{U}^d_{\vec{r}',\vec{n},\tau'}$ with $r_1 = r'_1 = 1$ satisfy $\varphi(\vec{\mathbf{u}},\tau) = \varphi(\vec{\mathbf{u}}',\tau')$. Suppose that $(\vec{r},\vec{n})$ and $(\vec{r}',\vec{n})$ both satisfy Assumption 4.7*

with the same $R \in \mathbb{N}_+$. Then, we have $\tau' \in \mathcal{C}S_{TT}^d(\tau)$ unless $\vec{\mathbf{u}}$ or $\vec{\mathbf{u}}'$ is located in some measure-zero subset of $\mathcal{U}_{\vec{r},\vec{n},\tau}^d$ or $\mathcal{U}_{\vec{r}',\vec{n},\tau'}^d$.

Proof. Suppose that $\tau' \notin \mathcal{C}S_{TT}^d(\tau)$. Then, there must exist three distinct indices $i_1, i_2, i_3 \in \{1, 2, \ldots, d\}$ such that (i_1, i_2, i_3) is of the correct order with respect to τ but is not of the correct order with respect to τ'. By Lemma B.1 and Lemma B.2, it holds almost surely that

$$(A.2) \qquad \operatorname{rank}\left(M'_{i_1,(i_2,i_3)}\right) \geq R^2 > \operatorname{rank}\left(M_{i_1,(i_2,i_3)}\right),$$

where $M_{i_1,(i_2,i_3)}$ and $M'_{i_1,(i_2,i_3)}$ are the matricization of $\varphi(\vec{\mathbf{u}},\tau)$ and $\varphi(\vec{\mathbf{u}}',\tau')$ defined in (3.4). However, it follows from $\varphi(\vec{\mathbf{u}},\tau) = \varphi(\vec{\mathbf{u}}',\tau')$ that $M_{i_1,(i_2,i_3)} = M'_{i_1,(i_2,i_3)}$, which contradicts (A.2). $\qquad\square$

Appendix B. Deferred proofs. We collect all deferred proofs in this section.

B.1. Proof of Theorem 4.8. The proof follows similar ideas and techniques as in the proof of Theorem 4.2. We also use two lemmas to establish the upper bound of the rank of $M_{i_1,(i_2,i_3)}$ and $M_{i_3,(i_1,i_2)}$, as well as the lower bound of the rank of $M_{i_2,(i_1,i_3)}$, given that (i_1, i_2, i_3) is of the correct order with respect to the underlying permutation τ.

LEMMA B.1. *Suppose that $i_s = \tau(j_s)$ for $s = 1, 2, 3$, where $1 \leq j_1 < j_2 < j_3 \leq d$. Then, for any $\vec{\mathbf{u}} \in \mathcal{U}_{\vec{r},\vec{n},\tau}^d$, it holds that*

$$(B.1) \qquad \operatorname{rank}(M_{i_1,(i_2,i_3)}) \leq \min_{j_1+1 \leq j \leq j_2} r_j$$

and that

$$(B.2) \qquad \operatorname{rank}(M_{i_3,(i_1,i_2)}) \leq \min_{j_2+1 \leq j \leq j_3} r_j.$$

Proof. We only prove (B.1) because (B.2) will hold by the same reasoning. Consider any $j \in \{j_1 + 1, j_1 + 2, \ldots, j_2\}$ and the matrix $\tilde{M} \in \mathbb{R}^{(n_{\tau(1)}n_{\tau(2)}\cdots n_{\tau(j-1)}) \times (n_{\tau(j)}n_{\tau(j+1)}\cdots n_{\tau(d)})}$ defined via

$$\begin{aligned}
\tilde{M}&\left((x_{\tau(1)}, x_{\tau(2)}, \ldots, x_{\tau(j-1)}), (x_{\tau(j)}, x_{\tau(j+1)}, \ldots, x_{\tau(d)})\right) \\
&= \tau(\vec{\mathbf{u}},\tau)(x_1, x_2, \ldots, x_d) \\
&= \mathbf{u}^{[\tau(1)]}(x_{\tau(1)})\mathbf{u}^{[\tau(2)]}(x_{\tau(2)})\cdots\mathbf{u}^{[\tau(j-1)]}(x_{\tau(j-1)}) \\
&\quad \cdot \mathbf{u}^{[\tau(j)]}(x_{\tau(j)})\mathbf{u}^{[\tau(j+1)]}(x_{\tau(j+1)})\cdots\mathbf{u}^{[\tau(d)]}(x_{\tau(d)})
\end{aligned}$$

for $1 \leq x_i \leq n_i$, $1 \leq i \leq d$. Note that

$$\mathbf{u}^{[\tau(1)]}(x_{\tau(1)})\mathbf{u}^{[\tau(2)]}(x_{\tau(2)})\cdots\mathbf{u}^{[\tau(j-1)]}(x_{\tau(j-1)}) \in \mathbb{R}^{1 \times r_j}$$

and

$$\mathbf{u}^{[\tau(j)]}(x_{\tau(j)})\mathbf{u}^{[\tau(j+1)]}(x_{\tau(j+1)})\cdots\mathbf{u}^{[\tau(d)]}(x_{\tau(d)}) \in \mathbb{R}^{r_j \times 1}.$$

We can know that $\tilde{M}$ is the product of a matrix of size $(n_{\tau(1)}n_{\tau(2)}\cdots n_{\tau(j-1)}) \times r_j$ and a matrix of size $r_j \times (n_{\tau(j)}n_{\tau(j+1)}\cdots n_{\tau(d)})$ and hence that $\operatorname{rank}(\tilde{M}) \leq r_j$. Therefore, it holds that $\operatorname{rank}(M_{i_1,(i_2,i_3)}) \leq r_j$ since $M_{i_1,(i_2,i_3)}$ is a submatrix of $\tilde{M}$. Then, (B.1) holds by taking the minimality of $j \in \{j_1 + 1, j_1 + 2, \ldots, j_2\}$. $\qquad\square$

LEMMA B.2. *Suppose that $i_s = \tau(j_s)$ for $s = 1, 2, 3$, where $1 \le j_1 < j_2 < j_3 \le d$, and that there exist R_1 and R_2 satisfying*

$$R_1 \le \min_{j_1+1 \le j \le j_2} r_j, \ R_2 \le \min_{j_2+1 \le j \le j_3} r_j$$

and

$$n_{i_1} \ge R_1, \ n_{i_2} \ge R_1 R_2, \ n_{i_3} \ge R_2.$$

Then, there exists a measure-zero subset $\Omega \in \mathcal{U}_{\vec{r},\vec{n},\tau}^d$, such that, for any $\vec{\mathbf{u}} \in \mathcal{U}_{\vec{r},\vec{n},\tau}^d \setminus \Omega$, it holds that

$$\mathrm{rank}(M_{i_2,(i_3,i_1)}) \ge R_1 R_2.$$

Proof. Similar to the proof of Lemma 4.5, one only needs to construct a $\vec{\mathbf{u}} \in \mathcal{U}_{\vec{r},\vec{n},\tau}^d$ such that the rank of the matrix M_1 associated with $\vec{\mathbf{u}}$ is at least $R_1 R_2$. Set

$$\mathbf{u}^{[i_1]}(x_{i_1}) = \begin{cases} e_{1,r_{j_1}} e_{q,r_{j_1+1}}^\top & \text{if } x_{i_1} = q, \ 1 \le q \le R_1, \\ 0_{r_{j_1} \times r_{j_1+1}} & \text{if } R_1 < x_{i_1} \le n_{i_1}, \end{cases}$$

$$\mathbf{u}^{[i_2]}(x_{i_2}) = \begin{cases} e_{p,r_{j_2}} e_{q,r_{j_2+1}}^\top & \text{if } x_{i_2} = (p-1)R_2 + q, \ 1 \le p \le R_1, \ 1 \le q \le R_2, \\ 0_{r_{j_2} \times r_{j_2+1}} & \text{if } R_1 R_2 < x_{i_2} \le n_{i_2}, \end{cases}$$

$$\mathbf{u}^{[i_3]}(x_{i_3}) = \begin{cases} e_{p,r_{j_3}} e_{1,r_{j_3+1}}^\top & \text{if } x_{i_3} = p, \ 1 \le p \le R_2, \\ 0_{r_{j_3} \times r_{j_3+1}} & \text{if } R_2 < x_{i_3} \le n_{i_3}, \end{cases}$$

and

$$\mathbf{u}^{[\tau(j)]}(y_{\tau(j)}) = \begin{cases} e_{1,r_j} e_{1,r_{j+1}}^\top & \text{if } 1 \le j < j_1, \\ \sum_{p=1}^{R_1} e_{p,r_j} e_{p,r_{j+1}}^\top & \text{if } j_1 < j < j_2, \\ \sum_{p=1}^{R_2} e_{p,r_j} e_{p,r_{j+1}}^\top & \text{if } j_2 < j < j_3, \\ e_{1,r_j} e_{1,r_{j+1}}^\top & \text{if } j_3 < j \le d. \end{cases}$$

Then, we can compute for $x_{i_1} = q_1$, $x_{i_2} = (p_2 - 1)R_2 + q_2$, $x_{i_3} = p_3$, where $1 \le q_1, p_2 \le R_1$, $1 \le q_2, p_3 \le R_2$, that

$$
\begin{aligned}
&M_{i_2,(i_3,i_1)}(x_{i_2},(x_{i_3},x_{i_1})) \\
&= M_{i_2,(i_3,i_1)}\left(x_{\tau(j_2)},(x_{\tau(j_3)},x_{\tau(j_1)})\right) \\
&= \prod_{j=1}^{j_1-1} \mathbf{u}^{[\tau(j)]}(y_{\tau(j)}) \cdot \mathbf{u}^{[\tau(j_1)]}(y_{\tau(j_1)}) \cdot \prod_{j=j_1+1}^{j_2-1} \mathbf{u}^{[\tau(j)]}(y_{\tau(j)}) \cdot \mathbf{u}^{[\tau(j_2)]}(y_{\tau(j_2)}) \\
&\quad \cdot \prod_{j=j_2+1}^{j_3-1} \mathbf{u}^{[\tau(j)]}(y_{\tau(j)}) \cdot \mathbf{u}^{[\tau(j_3)]}(y_{\tau(j_3)}) \cdot \prod_{j=j_3+1}^{d} \mathbf{u}^{[\tau(j)]}(y_{\tau(j)}) \\
&= \prod_{j=1}^{j_1-1} e_{1,r_j} e_{1,r_{j+1}}^\top \cdot e_{1,r_{j_1}} e_{q_1,r_{j_1+1}}^\top \cdot \prod_{j_1+1}^{j_2-1} \sum_{p=1}^{R_1} e_{p,r_j} e_{p,r_{j+1}}^\top \cdot e_{p_2,r_{j_2}} e_{q_2,r_{j_2+1}}^\top \\
&\quad \cdot \prod_{j=j_2+1}^{j_3-1} \sum_{p=1}^{R_2} e_{p,r_j} e_{p,r_{j+1}}^\top \cdot e_{p_3,r_{j_3}} e_{1,r_{j_3+1}}^\top \cdot \prod_{j=j_3+1}^{d} e_{1,r_j} e_{1,r_{j+1}}^\top \\
&= \left(e_{q_1,r_{j_1+1}}^\top \cdot \sum_{p=1}^{R_1} e_{p,r_{j_1+1}} e_{p,r_{j_2}}^\top \cdot e_{p_2,r_{j_2}}\right) \cdot \left(e_{q_2,r_{j_2+1}}^\top \cdot \sum_{p=1}^{R_2} e_{p,r_{j_2+1}} e_{p,r_{j_3}}^\top \cdot e_{p_3,r_{j_3}}\right) \\
&= \delta_{q_1,p_2} \delta_{q_2,p_3},
\end{aligned}
$$

which, combined with the fact that $M_{i_2,(i_3,i_1)}(x_{i_2},(x_{i_3},x_{i_1})) = 0$ as long as $x_{i_1} \leq R_1$, $x_{i_2} \leq R_1 R_2$, and $x_{i_3} \leq R_2$ do not hold simultaneously, yields that $\mathrm{rank}(M_{i_2,(i_3,i_1)}) = R_1 R_2$. So the proof is completed.   □

Then, one can prove Theorem 4.8.

Proof of Theorem 4.8. Since Assumption 4.7 holds for any (i_1,i_2,i_3) of the correct order with respect to τ, by Lemma B.1 and Lemma B.2, we have

$$\mathrm{rank}(M_{i_2,(i_3,i_1)}) \geq R^2 > \max\left\{\mathrm{rank}(M_{i_1,(i_2,i_3)}), \mathrm{rank}(M_{i_3,(i_1,i_2)})\right\}$$

as long as $\vec{\mathbf{u}}$ is not in some measure-zero set of $\mathcal{U}_{\vec{r},\vec{n},\tau}^d$. Thus, one can immediately conclude the correctness of Algorithm 3.3.   □

B.2. Proof of Theorem 4.16 and Corollary 4.17. The proof is similar to that of Theorem 4.11 and Corollary 4.12.

Proof of Theorem 4.16. Consider three different indices $i_1, i_2, i_3 \in \{1, 2, \ldots, d\}$, and assume that (i_1, i_2, i_3) is of the correct order with respect to τ. Denote E as the noise matrix in observing $M_{i_1,(i_2,i_3)}$, $M_{i_2,(i_3,i_1)}$, or $M_{i_3,(i_1,i_2)}$. With size at most $n_{\max} \times n_{\max}^2$, the matrix E by Theorem 4.13 satisfies that

$$\mathbb{P}\left(\sigma_{\max}(E) \geq \frac{\sigma}{2}\right) \leq \mathbb{P}\left(\sigma_{\max}(E) > (n_{\max} + \sqrt{n_{\max}} + t)\sigma_e\right) \leq 2e^{-t^2/2},$$

where $t = \frac{T}{2} - n_{\max} - \sqrt{n_{\max}}$. According to Theorem 4.14 (Weyl's inequality) and Assumption 4.15, the probability that Algorithm 3.3 obtains $\sigma_{R^2}(M_{i_1,(i_2,i_3)}) < \sigma/2$, $\sigma_{R^2}(M_{i_2,(i_1,i_3)}) > \sigma/2$, and $\sigma_{R^2}(M_{i_3,(i_1,i_2)}) < \sigma/2$ is at least

$$1 - 6 \cdot \exp\left(-\frac{t^2}{2}\right) = 1 - 6 \cdot \exp\left(-\frac{1}{8}(T - 2n_{\max} - 2\sqrt{n_{\max}})^2\right).$$

The proof is hence completed.   □

Proof of Corollary 4.17. Noticing that Algorithm 3.3 is called for $O(d \log d)$ times when implementing Algorithm 3.4, one can therefore conclude Corollary 4.17 from Theorem 4.16.   □

REFERENCES

[1] J. BALLANI AND L. GRASEDYCK, *Tree adaptive approximation in the hierarchical tensor format*, SIAM J. Sci. Comput., 36 (2014), pp. A1415–A1431.

[2] J. A. BENGUA, H. N. PHIEN, H. D. TUAN, AND M. N. DO, *Efficient tensor completion for color image and video recovery: Low-rank tensor train*, IEEE Trans. Image Process., 26 (2017), pp. 2466–2479.

[3] T. T. CAI AND A. ZHANG, *Rate-optimal perturbation bounds for singular subspaces with applications to high-dimensional statistics*, Ann. Statist., 46 (2018), pp. 60–89.

[4] S. CHEN, J. LI, Y. LI, AND A. R. ZHANG, *Learning polynomial transformations via generalized tensor decompositions*, in Proceedings of the 55th Annual ACM Symposium on Theory of Computing, 2023, pp. 1671–1684.

[5] Z. CHEN, Y. LI, AND J. LU, *Tensor ring decomposition: Optimization landscape and one-loop convergence of alternating least squares*, SIAM J. Matrix Anal. Appl., 41 (2020), pp. 1416–1442.

[6] S. CHENG, L. WANG, T. XIANG, AND P. ZHANG, *Tree tensor networks for generative modeling*, Phys. Rev. B, 99 (2019), 155131.

[7] A. CICHOCKI, N. LEE, I. OSELEDETS, A.-H. PHAN, Q. ZHAO, D. P. MANDIC, *Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions*, Found. Trends Mach. Learn., 9 (2016), pp. 249–429.

[8] A. CICHOCKI, A.-H. PHAN, Q. ZHAO, N. LEE, I. OSELEDETS, M. SUGIYAMA, D. P. MANDIC, *Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives*, Found. Trends Mach. Learn., 9 (2017), pp. 431–673.

[9] K. R. DAVIDSON AND S. J. SZAREK, *Local operator theory, random matrices and Banach spaces*, Handb. Geom. Banach Spaces, 1 (2001), pp. 317, 319–366.

[10] M.-S. DUPUY AND G. FRIESECKE, *Inversion symmetry of singular values and a new orbital ordering method in tensor train approximations for quantum chemistry*, SIAM J. Sci. Comput., 43 (2021), pp. B108–B131.

[11] N. HALKO, P.-G. MARTINSSON, AND J. A. TROPP, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.

[12] K. A. HALLBERG, *New trends in density matrix renormalization*, Adv. Phys., 55 (2006), pp. 477–526.

[13] M. HASHEMIZADEH, M. LIU, J. MILLER, AND G. RABUSSEAU, *Adaptive Learning of Tensor Network Structures*, preprint, arXiv:2008.05437, 2020.

[14] T. HIKIHARA, H. UEDA, K. OKUNISHI, K. HARADA, AND T. NISHINO, *Automatic structural optimization of tree tensor networks*, Phys. Rev. Res., 5 (2023), 013031.

[15] S. HOLTZ, T. ROHWEDDER, AND R. SCHNEIDER, *The alternating linear scheme for tensor optimization in the tensor train format*, SIAM J. Sci. Comput., 34 (2012), pp. A683–A713.

[16] R. G. JHA, *Tensor Renormalization of Three-Dimensional Potts Model*, preprint, arXiv:2201.01789, 2022.

[17] Y. KHOO, J. LU, AND L. YING, *Efficient construction of tensor ring representations from sampling*, Multiscale Model. Simul., 19 (2021), pp. 1261–1284.

[18] C. LI AND Z. SUN, *Evolutionary topology search for tensor network decomposition*, in International Conference on Machine Learning, PMLR, 2020, pp. 5947–5957.

[19] C. LI, J. ZENG, C. LI, C. F. CAIAFA, AND Q. ZHAO, *Alternating local enumeration (TnALE): Solving tensor network structure search with fewer evaluations*, in International Conference on Machine Learning, PMLR, 2023, pp. 20384–20411.

[20] C. LI, J. ZENG, Z. TAO, AND Q. ZHAO, *Permutation search of tensor network structures via local sampling*, in International Conference on Machine Learning, PMLR, 2022, pp. 13106–13124.

[21] Y. LIU, Y. LU, W. OU, Z. LONG, AND C. ZHU, *Adaptively Topological Tensor Network for Multi-View Subspace Clustering*, preprint, arXiv:2305.00716, 2023.

[22] C. LUBICH, T. ROHWEDDER, R. SCHNEIDER, AND B. VANDEREYCKEN, *Dynamical approximation by hierarchical Tucker and tensor-train tensors*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 470–494.

[23] V. MURG, F. VERSTRAETE, R. SCHNEIDER, P. R. NAGY, AND O. LEGEZA, *Tree tensor network state with variable tensor order: An efficient multireference method for strongly correlated systems*, J. Chem. Theory Comput., 11 (2015), pp. 1027–1036.

[24] C. NIE, H. WANG, AND L. TIAN, *Adaptive tensor networks decomposition*, in British Machine Vision Conference, 2021, pp. 1–14.

[25] R. ORÚS, *A practical introduction to tensor networks: Matrix product states and projected entangled pair states*, Ann. Phys., 349 (2014), pp. 117–158.

[26] R. ORÚS, *Tensor networks for complex quantum systems*, Nature Rev. Phys., 1 (2019), pp. 538–550.

[27] I. V. OSELEDETS, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317.

[28] Y. REN, H. ZHAO, Y. KHOO, AND L. YING, *High-dimensional density estimation with tensorizing flow*, Res. Math. Sci., 10 (2023), 30.

[29] M. RUDELSON AND R. VERSHYNIN, *Non-asymptotic theory of random matrices: Extreme singular values*, in Proceedings of the International Congress of Mathematicians 2010 (ICM 2010) (In 4 Volumes) Vol. I: Plenary Lectures and Ceremonies Vols. II–IV: Invited Lectures, World Scientific, 2010, pp. 1576–1602.

[30] U. SCHOLLWÖCK, *The density-matrix renormalization group in the age of matrix product states*, Ann. Phys., 326 (2011), pp. 96–192.

[31] Y.-Y. SHI, L.-M. DUAN, AND G. VIDAL, *Classical simulation of quantum many-body systems with a tree tensor network*, Phys. Rev. A, 74 (2006), 022320.

[32] R. SOCHER, D. CHEN, C. D. MANNING, AND A. NG, *Reasoning with neural tensor networks for knowledge base completion*, Adv. Neural Inf. Process. Syst., 26 (2013), pp. 926–934.

[33] E. STOUDENMIRE AND D. J. SCHWAB, *Supervised learning with tensor networks*, Adv. Neural Inf. Process. Syst., 29 (2016), pp. 4799–4807.

[34] S. SZALAY, M. PFEFFER, V. MURG, G. BARCZA, F. VERSTRAETE, R. SCHNEIDER, AND Ö. LEGEZA, *Tensor product methods and entanglement optimization for ab initio quantum chemistry*, Int. J. Quantum Chem., 115 (2015), pp. 1342–1391.

[35] L. N. TREFETHEN AND D. BAU III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[36] L. R. TUCKER, *Implications of factor analysis of three-way matrices for measurement of change*, Probl. Measuring Change, 15 (1963), pp. 122–137.

[37] F. VERSTRAETE AND J. I. CIRAC, *Renormalization Algorithms for Quantum-Many Body Systems in Two and Higher Dimensions*, preprint, arXiv:cond-mat/0407066, 2004.

[38] F. VERSTRAETE AND J. I. CIRAC, *Matrix product states represent ground states faithfully*, Phys. Rev. B, 73 (2006), 094423.

[39] W. WANG, V. AGGARWAL, AND S. AERON, *Efficient low rank tensor ring completion*, in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 5697–5705.

[40] W. WANG, Y. SUN, B. ERIKSSON, W. WANG, AND V. AGGARWAL, *Wide compression: Tensor ring nets*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9329–9338.

[41] F.-Y. WU, *The Potts model*, Rev. Modern Phys., 54 (1982), 235.

[42] L. YUAN, J. CAO, X. ZHAO, Q. WU, AND Q. ZHAO, *Higher-dimension tensor completion via low-rank tensor ring decomposition*, in 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), IEEE, 2018, pp. 1071–1076.

[43] L. YUAN, C. LI, D. MANDIC, J. CAO, AND Q. ZHAO, *Tensor ring decomposition with rank minimization on latent space: An efficient approach for tensor completion*, in Proceedings of the AAAI Conference on Artificial Intelligence, 33 (2019), pp. 9151–9158.

[44] X. YUAN, J. SUN, J. LIU, Q. ZHAO, AND Y. ZHOU, *Quantum simulation with hybrid tensor networks*, Phys. Rev. Lett., 127 (2021), 040501.

[45] Q. ZHAO, G. ZHOU, S. XIE, L. ZHANG, AND A. CICHOCKI, *Tensor Ring Decomposition*, preprint, arXiv:1606.05535, 2016.

[46] Y. ZHOU, A. R. ZHANG, L. ZHENG, AND Y. WANG, *Optimal high-order tensor SVD via tensor-train orthogonal iteration*, IEEE Trans. Inform. Theory, 68 (2022), pp. 3991–4019, https://doi.org/10.1109/TIT.2022.3152733.