Permutation Search of Tensor Network Structures via Local Sampling

Chao Li† 1 Junhua Zeng † 2 1 Zerui Tao 3 1 Qibin Zhao 1

Abstract
Recent works put much effort into tensor network structure search (TN-SS), aiming to select suitable tensor network (TN) structures, involving the TN-ranks, formats, and so on, for the decomposition or learning tasks. In this paper, we consider a practical variant of TN-SS, dubbed TN permutation search (TN-PS), in which we search for good mappings from tensor modes onto TN vertices (core tensors) for compact TN representations. We conduct a theoretical investigation of TN-PS and propose a practically-efficient algorithm to resolve the problem. Theoretically, we prove the counting and metric properties of search spaces of TN-PS, analyzing for the first time the impact of TN structures on these unique properties. Numerically, we propose a novel meta-heuristic algorithm, in which the searching is done by randomly sampling in a neighborhood established in our theory, and then recurrently updating the neighborhood until convergence. Numerical results demonstrate that the new algorithm can reduce the required model size of TNs in extensive benchmarks, implying the improvement in the expressive power of TNs. Furthermore, the computational cost for the new algorithm is significantly less than that in (Li & Sun, 2020).

1. Introduction
Over the years, tensor network (TN) has been widely applied to various technical fields. It enables us to resolve extremely high-dimensional problems, such as deep learning (Novikov et al., 2015; Kossaifi et al., 2020), probability density estimation (Glasser et al., 2019; Miller et al., 2021; Novikov et al., 2021), partial differential equations (Bachmayr et al., 2016; Richter et al., 2021), and quantum circuit simulation (Markov & Shi, 2008; Huggins et al., 2019), with acceptable computational and storage costs. However, as an inevitable side-effect, practitioners have to face a notorious challenge when applying TNs to practical tasks: how to efficiently select the optimal TN structures from a massive quantity of candidates?

Recent works thus put effort into this challenge, in the heading of TN structure search (TN-SS). Most recently, there have been studies, which focus on searching TN ranks (Hashemizadeh et al., 2020; Kodryan et al., 2020), formats (Hayashi et al., 2019; Li & Sun, 2020), and orders (Li et al., 2020; Qiu et al., 2021) to achieve more compact representations. These results also confirm numerically that the structures impact the expressive power (Cohen et al., 2016) of TNs in learning tasks.

In this paper, we consider a practical variant of TN-SS. The goal is to improve the compactness and expressiveness of a TN while preserving its format, such as tensor train (TT, Oseledets 2011) or tensor ring (TR, Zhao et al. 2016). Several existing works (Zhao et al., 2016; Zheng et al., 2021) have noticed that the mapping from tensor modes onto the TN vertices, also known as core tensors, also influences the expressive power of the model. To see this, we implement a toy experiment, in which TR is utilized to approximate a tensor of order four. Figure 1 shows the required ranks for achieving the same approximation accuracy in the experiment. We see that a good “mode-vertex” mapping (corresponding to Model 1) would produce smaller ranks than the other two models. It implies lower computational and storage costs and more promising generalization capability in learning tasks (Khavari & Rabusseau, 2021). This fact thus motivates this work for searching both the optimal TN-ranks and “mode-vertex” mappings.

Despite the potential benefit, searching for the optimal “mode-vertex” mappings is non-trivial in general. For instance, there would be \( O(N!) \) different candidates for TR of order \( N \geq 3 \) even though the optimal ranks are known. It is apparently unacceptable to solve it by exhaustive search, particularly when combinatorially searching the TN-ranks.
is required as well. Unlike TN-SS, it also appears new theoretical questions for the variant: how fast the scale of the mappings grows with the structures parameters, e.g., format, order and ranks? Why does the growth rate change? And what bounds the growth?

To this end, we conduct a thorough investigation of this variant, named TN permutation search (TN-PS), from both the theoretical and numerical aspects. Theoretically, we answer the preceding questions by analyzing the counting property of the search space of TN-PS, proving a universal non-asymptotic bound for TNs in arbitrary formats. The result is helpful for fast estimation of the computational budget in searching. We also establish the basic geometry for TN-PS with group-theoretical instruments, involving the (semi-)metric and neighborhood of the search space, such that the local searching can be applied to the task.

Numerically, we develop an efficient algorithm for TN-PS. In contrast to the existing sampling-based methods (Hayashi et al., 2019; Li & Sun, 2020), we draw samples in the established neighborhood to explore the “steepest-descent” path of the landscape, thereby accelerating the searching procedure and decreasing the computational cost. Experimental results on extensive benchmarks demonstrate that the proposed algorithm is unique to resolving TN-PS consistently so far, and the required model evaluations are much fewer than the previous algorithm. We summarize the main contributions of this work as follows:

- We propose for the first time the problem of tensor-network permutation search (TN-PS), an important variant of TN-SS in practice;
- We rigorously prove new theoretical properties for TN-PS, involving the counting, (semi-)metric, and neighborhood, revealing how TN structures impact these properties;
- We develop a local-sampling-based meta-heuristic, which significantly reduces the computational cost compared to (Li & Sun, 2020).

1.1. Related Works

Searching tensor-network (TN) structures. Searching the optimal structures for TNs is typically thought of as an extension of the rank selection problem for tensor learning (Zhao et al., 2015; Yokota et al., 2016; Zheng et al., 2016; Cheng et al., 2020; Mickelin & Karaman, 2020; Cai & Li, 2021; Hawkins & Zhang, 2021; Li et al., 2021; Long et al., 2021; Sedighin et al., 2021), which is widely known to be challenging, especially when the TN formats contain cycles (Landsberg et al., 2011; Batselier, 2018; Ye & Lim, 2019). More recently, several studies put much effort into this problem, i.e., TN-SS, for exploring unknown formats (Hayashi et al., 2019; Hashemizadeh et al., 2020; Kodryan et al., 2020; Li & Sun, 2020; Nie et al., 2021).

The latest works (Razin et al., 2021; 2022) also study the implicit regularization over TN-ranks. Another line of work closely related to ours is those that study the partition issue for hierarchical Tucker (HT, Falció et al. 2020; Haberstich et al. 2021) decomposition, which aims to search for the optimal tree structures. Compared to these works, we focus on the search over the “mode-vertex” mappings, which have remained unexplored until now.

Sampling-based optimization. Our new algorithm is inspired by zeroth-order optimization, which is also known as gradient-free optimization or bandit optimization. The methods can date back to stochastic hill-climbing (Russell & Norvig, 1995), followed by numerous evolutionary programming algorithms (Back, 1996), and are restudied recently by Golovin et al. (2019) and applied to various machine learning tasks (Liu et al., 2020a;b; Savarese et al., 2021; Singh, 2021). Inspired by the work (Golovin et al., 2019), we refine the sampling strategy for TN-PS by taking the unique property of the neighborhood into account but maintaining its original simplicity and efficiency.

2. Preliminaries

We first summarize notations and elementary results used throughout the paper. After that, definitions related to tensor networks (TNs) are reviewed for the self-contained purpose.

Throughout the paper, we use blackboard letters, such as $G$ and $S$, to denote sets of subjects. With additional structures, they are also used to represent specific algebraic subjects according to the context, such as groups, fields or linear spaces. In particular, we use $S_N$, $\mathbb{R}$, $\mathbb{Z}^+$, and $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ to represent the symmetric group of order $N$, the real field, positive integers and the real linear space of dimension $I_1 \times I_2 \times \cdots \times I_N$, respectively. The size of a finite set $A$ is denoted by $|A|$, and the Cartesian product of two sets $A$ and $B$ is denoted by $A \times B$. We say two sets, e.g., $A$, $B$, are equivalent if there exists a bijective mapping from $A$ and $B$, and sometimes write $A = B$ without explicit declaration of
the mapping if it is unambiguous. For convenience, we use \([N] \subseteq \mathbb{Z}^+\) to denote a set of positive integers from 1 to \(N\), where \(\subseteq\) represents the subset relation.

A graph \(G = (V, E)\) consists of a vertex set \(V\) and an edge set \(E\). For a graph \(G\) of \(N\) vertices, the set of its automorphisms, written \(\text{Aut}(G)\), is a collection of vertex permutations, under which the edges are preserved, and equivalent to a subgroup of the symmetric group, i.e., \(\text{Aut}(G) \leq S_N\). We call \(r \in \mathbb{R}\) be a (spanning) subgraph of \(G\) if \(V_H = V\) and \(E_H \subseteq E\). Let \(K_N = (V, E_{K_N})\) be a complete graph with \(N\) vertices and \(G_N\) be the set containing all subgraphs of \(K_N\). We then know that any simple graphs of \(N\) vertices are elements of \(G_N\). The \textit{minimum and maximum degree} of a graph \(G\) are denoted by \(\delta\) and \(\Delta\), respectively.

2.1. Tensor and Tensor Networks (TNs)

We consider an \textit{order}-\(N\) tensor as a multidimensional array of real numbers represented by \(X_{i_1,\ldots,\,i_N} \in \mathbb{R}^{I_1 \times \ldots \times I_N}\), where the indices \(i_n, n \in [N]\) correspond to the \(I_n\)-associated tensor mode. Sometimes we ignore the indices by representing the same tensor as \(X\) for notational simplicity. \textit{Tensor contraction} roughly refers to the process of summing over a pair of repeated indices between two tensors, which is thought of as a natural extension of matrix multiplication into high-order tensors. An explicit calculation of tensor contraction used in this paper follows the definition in (Cichocki et al., 2016).

We consider tensor network (TN) as defined by Ye & Lim (2019). Suppose a sequence of vector spaces \(\mathbb{R}^{I_1}, i \in [N]\) and an edge-labelled simple graph \((G, r) = (V, E, r)\), where \(r : E \rightarrow \mathbb{Z}^+\) represents the function labelling edges with positive integers. TN is thus intuitively defined as a set of tensors, whose elements are of the form of a sequence of tensor contraction of “core tensors” corresponding to vertices of \(G\). See (Ye & Lim, 2019) for an explicit definition of a TN. In the paper we refer to those core tensors as \textit{vertices}, to the unlabelled graph \(G\) as \textit{TN format}, and to the function \(r\) as \textit{TN-(model)-ranks}. Being consistent with (Ye & Lim, 2019), we use the same mathematical expression \(\text{TN}(G, r, \mathbb{R}^{I_1}, \mathbb{R}^{I_2}, \ldots, \mathbb{R}^{I_N})\) to represent a TN in our analysis. The expression is also rewritten as \(\text{TN}(G, r)\) for shorthand if \(\mathbb{R}^{I_n}, n \in [N]\) are unimportant in the context. Let \(\mathbb{F}_G\) be the set consisting of all possible functions of \(r\)’s associated to \(G\). Then note that \(\mathbb{F}_G\) is equivalent to a positive cone except zero of dimension \(|E|\), i.e., \(\mathbb{F}_G = \mathbb{Z}^+ \langle |E| \rangle\).

2.2. TN Structure Search (TN-SS)

Let \(X \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}\) be an order-\(N\) tensor. TN-SS \textit{without noise} is to solve an optimization problem as follows:

\[
\min_{r \in \mathbb{F}_{K_N}} \phi(K_N, r), \quad \text{s.t. } X \in \text{TN}(K_N, r),
\]  

where \(\phi : \mathbb{G}_N \times \mathbb{F}_{K_N} \rightarrow \mathbb{R}\) represents a loss function measuring the model complexity of a TN. Note that, although in (1) the first term of \(\phi\) is fixed to be \(K_N\), the TN format can degenerate into any simple graphs of \(N\) vertices, as the edges of labeling with “1”, i.e., \(\{e \in E_{K_N} | r(e) = 1\}\), can be harmlessly discarded from the format (Ye & Lim, 2019; Hashemizadeh et al., 2020). We see that solving (1) is an \textit{integer programming} problem, generally NP-complete (Papadimitriou & Yannakakis, 1982). Nevertheless, thanks to the fact \(\mathbb{F}_{K_N} = \mathbb{Z}^+ |E_{K_N}|\), some practical algorithms have been proposed (Hashemizadeh et al., 2020; Kodryan et al., 2020; Li & Sun, 2020), as \(\mathbb{Z}^+ |E_{K_N}|\) is a well-defined metric space with the isotropic property. However, we will see next that such good properties do not hold for TN-PS anymore in general.

3. Tensor-Network Permutation Search (TN-PS)

In this section, we first make precise the problem of TN-PS and then prove the properties involving counting, metric, and neighborhood, which are crucial for both understanding the problem and deriving efficient algorithms.

3.1. Problem Setup

Recall the example illustrated in Figure 1. Suppose a tensor \(X\) of order \(N\) and a simple graph \(G_0\), dubbed \textit{template}, of \(N\) vertices. Apart from the TN-ranks, the primary goal of TN-PS is to find the optimal mappings in some sense from the modes of \(X\) onto vertices of \(G_0\). We thus easily see that the problem amounts to searching the optimal \textit{permutation} of vertices of a graph. More precisely, solving TN-PS is to repeatedly index the vertices of \(G_0\) consecutively from 1 to \(N\), and then to seek the optimal index sequence in some sense from all possibilities. Since the permutations are bijective to each other, the TN structures arising from these permutations naturally form an equivalence class, of which all elements preserve the same “diagram” as \(G_0\). Formally, such the equivalence class to the template \(G_0 = (V, E_0)\) can be written as follows:

\[
\mathbb{G}_0 = \{ G \in \mathbb{G}_N | G \cong G_0 \},
\]  

where \(\cong\) denotes the relation of graph \textit{isomorphism}, meaning that for each \(G \in \mathbb{G}_0\) there exists a vertex permutation \(g_G \in S_N\) such that \(G = (g_G(V), E_0)\) holds, or \(G = g_G \cdot G_0\) for shorthand. TN-PS (without noise) is thus defined by restricting the search space of (1) to \(\mathbb{G}_0\) as follows:

\[
\min_{(G, r) \in \mathbb{G}_0 \times \mathbb{F}_{G_0}} \phi(G, r), \quad \text{s.t. } X \in \text{TN}(G, r).
\]  

Compared to TN-SS, we search TN structures from a new space consisting of two ingredients: a non-trivial graph set
Figure 2. “Geometrical shape” of search spaces of TN-SS and TN-PS, where the equivalence class \( G_0 \) makes the “shape” for TN-PS as a combination of flips of low-dimensional spaces.

\( G_0 \) and \( \mathbb{Z}^{+,|G_0|} \) that corresponds to the TN-ranks. We see that TN-SS is no longer an integer programming problem as TN-PS due to the irregular geometry of \( G_0 \). Meanwhile, the size of the new search space varies with different template \( G_0 \). Figure 2 visualizes intuitively the “geometrical shape” of the search space for TN-PS associated to a template of three vertices and two edges. We see that the search space of TN-PS is more “collapsed” than the original TN-SS. One immediate consequence of collapsing is that the searching path and solutions for TN-SS would run out of the TN-PS region, thereby failing to preserve the original TN format. Next, we will establish formal statements for these observations, and the results will help develop feasible algorithms for resolving TN-PS.

3.2. Counting TN Structures

We begin by counting the size of the new search space, proving that the graph degrees of the template \( G_0 \) give a universal bound for the size of the search space of TN-PS.

Suppose first a simple graph \( G_0 = (\mathcal{V}, \mathcal{E}_0) \) of \( N \) vertices as the template, by which we then construct the set \( G_0 \) as Eq. (2). As mentioned above, we have known two facts: 1) \( \text{Aut}(G_0) \) forms a subgroup of \( S_N \), i.e., \( \text{Aut}(G_0) \subseteq S_N \), such that \( G_0 = a \cdot G_0 \) for any \( a \in \text{Aut}(G_0) \); and 2) for every \( G \in G_0 \) there exists \( g_G \in S_N \) such that \( G = g_G \cdot G_0 \). By these facts, \( G = g_G \cdot G_0 \) holds for any \( g_G \in S_N \), implying that for each \( G \in G_0 \) there exists a left coset of \( \text{Aut}(G_0) \), which is of the form \( g_G \cdot \text{Aut}(G_0) := \{g_G \cdot a | a \in \text{Aut}(G_0)\} \). According to the Lagrange’s theorem in group theory, we thus obtain the following equation with respect to the size of \( G_0 \):

\[
|S_N| = |G_0| \cdot |\text{Aut}(G_0)|.
\]

Table 1 lists the values of \( |\text{Aut}(G_0)| \) associated with several commonly used TNs. The size of \( G_0 \) for those TNs can be therefore derived by (4), shown in the last row of Table 1.

However, counting the automorphisms for a general graph is difficult (Chang et al., 1995). Blow we prove that the size of the search space of TN-PS is controlled by the the minimum and maximum degree of \( G_0 \). For convenience, we further assume that TN-ranks are only searched within a finite range \( \mathbb{F} \subset \mathbb{Z}^{+,|G_0|} \), meaning that the rank \( r(e) \leq R \) holds for any \( e \in \mathbb{F} \). We then have the following counting bounds.

**Theorem 3.1.** Assume \( G_0 \) to be a simple and connected graph of \( N \) vertices, and \( G_0 \) is constructed as (2). Let \( \delta = N/d_1 \) and \( \Delta = N/d_2 \), \( d_1 \geq d_2 > 1 \), be the minimum and maximum degree of \( G_0 \), respectively. The size of the search space of (3), written \( L_{G_0,R} := G_0 \times \mathbb{F} \), is bounded as follows:

\[
R^{\frac{d_1^2}{2} - 1} \cdot N! \geq |L_{G_0,R}| \geq R^{\frac{d_2^2}{2} - 1} \cdot e^{-\gamma(d_2)} \cdot N \cdot \frac{1}{2} \log d_2 - 1/24,
\]

where \( \gamma(d) = \log d + \frac{1}{d} - 1 \) is a positive and monotonically increasing function for \( d > 1 \).

Proving the above theorem requires the following lemma about an upper-bound of the size of \( \text{Aut}(G_0) \), of which the proof is given in Appendix A.

**Lemma 3.2.** Let \( G_0 \) be a simple graph of \( N \) vertices, and \( \text{Aut}(G_0) \) be the set containing automorphisms of \( G_0 \). Assume that \( G_0 \) is connected and its maximum degree \( \Delta \) satisfies \( N/\Delta = d > 1 \), then the inequality

\[
|\text{Aut}(G_0)| \leq N! \cdot e^{-\gamma(d)} \cdot N \cdot \frac{1}{2} \log d_2 - 1/24
\]

holds, where \( \gamma(\cdot) \) is defined in Theorem 3.1.

As shown in (5), the bounds of \( |L_{G_0,R}| \) are determined by three factors: the number of vertices \( N \), the searching range of TN-ranks \( R \), and the graph degrees of \( G_0 \) parameterized by \( d_1 \) and \( d_2 \). Figure 3 shows the bounds in (5) with varying these factors. We see from the left panel that the upper and lower bounds go closer with increasing the value of \( d \) (where we assume \( d_1 = d_2 = d \) for brevity). It implies that the bounds are tight for graphs with small degrees. We also see from the middle panel that \( |L_{G_0,R}| \) grows fast with \( N \), even though the graph degree \( d \) has been sufficiently small such as in TT/TR, while the growth is relatively slow with increasing \( R \), the search range for TN-ranks.
3.3. Semi-Metric and Neighborhood

The notion of metric and neighborhood of the search space are fundamental for most steepest-descent-based optimization methods. We have seen that they are well-defined for TN-SS but remain unknown for TN-PS. To address the issue, we establish below a new (semi-)metric and neighborhood for TN-PS with rigorous proofs using group-theoretic instruments. The application of these results to developing efficient algorithms for TN-PS will be introduced in the next section.

We begin by establishing the semi-metric, a relaxation of metric satisfying separation and symmetry except possibly for the triangle inequality, over the graph set \( G_0 \). Although there has been much literature in which different definitions of the graph metric or similarity are proposed, most of them are computationally hard (Koutra et al., 2011). Unlike those works, we construct the semi-metric over \( G_0 \) based on the equivalence property of its elements given in (2), so it can be built up by graph isomorphisms in a simple fashion.

Recall the symmetric group \( S_N \). Let \( T_N \subseteq S_N \) be the set consisting of its all adjacent transpositions, the operations of swapping adjacent two integers in \([N]\) and fixing all other integers. We thus know from group theory that \( T_N \) generates \( S_N \). Furthermore, let \( d_{T_N} : S_N \times S_N \to R \) be the word metric (Lück, 2008) of \( S_N \) induced by \( T_N \). Intuitively, the value of \( d_{T_N}(p_1, p_2) \), \( p_1, p_2 \in S_N \) reflects the minimum number of adjacent swapping operations required for transforming the permutation from \( p_1 \) to \( p_2 \). Since we saw in Section 3.1 that for each \( G \in G_0 \) there is a permutation \( g_g \in S_N \) such that \( G = g_g \circ G_0 \), we thus construct a function \( d_{G_0} : G_0 \times G_0 \to R \) using the word metric \( d_{T_N} \) as follows:

\[
d_{G_0}(G_1, G_2) = \min_{p_1, g_1 \in Aut(G_0), i=1,2} d_{T_N}(p_1, p_2), \quad (7)
\]

where \( G_1, G_2 \in G_0 \) and \( g_1, g_2 \in S_N \) are permutations satisfying \( G_i = g_i \circ G_0, \quad i = 1, 2 \). The following lemma shows that (7) is in fact a semi-metric function, followed by the construction of the corresponding neighborhood in \( G_0 \).

**Lemma 3.3.** Let \( G_0 \) be a simple graph and \( G_0 \) be the set defined as (2). The function \( d_{G_0} : G_0 \times G_0 \to R \) defined by (7) is a semi-metric on \( G_0 \). Furthermore, let \( \mathbb{I}_d(G) \) be a set constructed as follows:

\[
\mathbb{I}_d(G) = \{ G' \in G_0 | G' = q \prod_{i=1}^{d} t_i \circ G_0, \quad \text{ s.t. } G_0, t_i \in T_N, i \in [d] \} \quad (8)
\]

Then \( G_0 \) is the neighborhood of \( G = g \circ G_0 \in G_0 \) induced by (7), with the radius \( D \in \mathbb{Z}^+ \cup \{0\} \).

We see from (8) that \( G_0 \) consists of combinations of two sets: \( Aut(G_0) \) and \( T_N \), followed by the permutation representative \( g \) associated to the center graph \( G \). It thus suggests a straightforward sampling method over \( G_0 \), that is, combinatorially sampling over \( Aut(G_0) \) and \( T_N \) from some distributions. However, obtaining all elements of \( Aut(G_0) \) is computationally hard (NP-intermediate, Goldwasser et al. 1989) in general. To avoid this, we prove that sampling using Alg. 1 can cover all elements of \( G_0 \) without sampling directly over \( Aut(G_0) \).

**Theorem 3.4.** For every \( G' \in \mathbb{I}_d(G) \) with \( G \in G_0 \) and \( d \geq 1 \), the probability that the output of Alg. 1 equals \( G' \) is positive.

The (semi-)metric and neighborhood for the overall search space of TN-PS, i.e., \( G_0 \times F_{G_0,R} \), can be thus derived by composing the Euclidean metric of \( F_{G_0,R} \subseteq \mathbb{R}^{|E_0|} \). In the next section, Alg. 1 will be applied to the new algorithm, by which the searching efficiency is significantly improved.

4. Meta-Heuristic via Local Sampling

We present now a new meta-heuristic algorithm for searching TN structures. Unlike the existing methods such as (Li & Sun, 2020), we exploit the information of the “steepest-descent” direction, estimated by sampling over a neighborhood of the search space, to accelerate the searching procedure.

Suppose a tensor \( \mathcal{X} \) of order \( N \). For the practical purpose, we take the influence of noise into (3), which is given by

\[
\min_{G,r,z} \phi(G,r) + \lambda \cdot RSE(\mathcal{X}, Z)
\]

s.t. \( (G,r) \in G_0 \times F_{G_0,R} \), and \( Z \in TNS(G,r) \)
where $\lambda > 0$ denotes a tuning parameter associated with the noise variance, and $RSE$ is the function of relative squared error (RSE) for modeling the influence of Gaussian noise. In other applications such as in generative models (Liu et al., 2021), it can be replaced by KL-divergence without modifying the algorithm details. The searching algorithm is illustrated in Alg. 2, where $N_{[R]}(a, b)$ denotes a Gaussian distribution of the mean $a$ and variance $b$, followed by a truncation operation such that the samples out of the range $[R]$ are pulled back to the closest bound, $Ber(p)$ denotes the Bernoulli distribution of the mean $p$, and $I$ denotes the identity matrix of dimension $|E_0| \times |E_0|$. In each iteration, we elaborate the algorithm into three phases: local-sampling, evaluation, and updating. Suppose the starting point $(G^{(m)}, r^{(m)})$ at the $m$th iteration. In the local-sampling phase, we randomly draw samples for both the TN-ranks ($s_k$) and the “mode-vertex” maps ($H_k$) over the neighborhood centered at $(G^{(m)}, r^{(m)})$. The aim is to explore good descent directions within the neighborhood. The sampling distributions, involving the rounded truncated Gaussian and the uniform distribution in Alg. 1, are chosen as a non-informative searching prior. Note that for the two variance-related parameters $c_1, c_2 \in [0.9, 1)$, we apply the annealing trick to the shrinkage of the sampling range in each iteration. The trick guarantees the convergence of the algorithm. In the evaluation phase, we employ arbitrarily proper optimization or learning methods to minimize RSE or other alternatives for the sampled structures $(H_k, s_k), k \in [\#Sample]$. In the updating phase, we calculate the overall loss function $f_k$ for each sampled structures, and then update $(G^{(m+1)}, r^{(m+1)})$ once there exists new samples, whose performance is better than $(G^{(m)}, r^{(m)})$. More precisely, let $f_0$ be the loss of (9) with respect to $(G^{(m)}, r^{(m)})$ and $f_{min}$ be the minimum among all $f_k$’s. If $f_{min} < f_0$, we update $(G^{(m+1)}, r^{(m+1)}) = (H_{min}, s_{min})$; otherwise, we remain $(G^{(m+1)}, r^{(m+1)}) = (G^{(m)}, r^{(m)})$.

**Discussion.** Compared with the global-sampling methods, such as genetic algorithm (GA) (Hayashi et al., 2019; Li & Sun, 2020), we restrict the sampling range into the neighborhoods of the structure candidates, rather than the whole search space. The advantages in doing so are mainly two-folds: first, the neighborhood geometry allows to construct gradient-like directions, which result in a faster decrease of the loss function if the landscape is related smooth; second, a smaller sampling range can mitigate the curse of dimensionality. Otherwise, the algorithm would “lose itself” in searching if the TN structure is large scale. However, it should be mentioned that the local-sampling methods would perform worse than the global-searching ones if the landscape is too “flat” or “swinging”. We conjure with rich empirical observations that TN-PS (including TN-SS) seems more suitable for “local-sampling” methods. Since a small perturbation on the structure such as ranks would not dramatically change the RSE, it implies that the landscape of (9) tends to be smooth. Although a rigorous discussion on this issue remains open, we use extensive numerical results to verify the efficiency of Alg. 2 in the next section.

**5. Experimental Results**

In this section, we numerically verify the effectiveness and efficiency of the proposed method on both the synthetic and real-world tensors.

**5.1. Synthetic Data in TT/TR Format and Beyond**

Using synthetic data, we first verify: (a) TN-PS can reduce the required TN model size for the low-rank tensor approximation task, reflecting the improvement of the expressive power of TNs; and (b) the proposed local-sampling method achieves more efficient searching than the existing
Table 2. Experimental results on synthetic data in TT/TR format. In the table, Eff. denotes the parameter efficiency defined in (Li & Sun, 2020), #Eva. denotes the number of evaluations, and “-” denotes that the methods fail to satisfy the condition $RSE \leq 10^{-4}$.

| Trial | TR-SVD | TR-LM | TR-ALSAR | Bayes-TR | Greedy | TNGA⁺ | TNGA⁺ methods |
|-------|--------|-------|----------|----------|--------|-------|----------------|
|       |        |       |          |          |        |       |                |
| Order 4 – Eff. ↑ (Is TT/TR format preserved? Yes or No) #Eva. ↓ |
| A     | 1.00   | 1.00  | 0.21     | 1.00     | 1.00   | 1.00  | 1.00 [1000]    |
| B     | 0.64   | 1.00  | 1.00     | 0.64     | 1.00   | 0.89  | 1.00 [1000]    |
| C     | 1.17   | 1.17  | 0.23     | 1.00     | 1.17   | 1.17  | 1.17 [1000]    |
| D     | 0.57   | 0.57  | 0.32     | -        | 1.00   | 1.00  | 1.00 [1000]    |
| E     | 0.43   | 0.48  | 0.40     | 0.40     | 1.00   | 1.00  | 1.00 [1000]    |

| Order 6 – Eff. ↑ (Is TT/TR format preserved? Yes or No) #Eva. ↓ |
| A     | 0.21   | 0.44  | -        | -        | 0.16   | 0.82  | 1.00 [1500]    |
| B     | 0.14   | 0.15  | 0.14     | -        | 0.27   | -     | 1.00 [1350]    |
| C     | 0.57   | 1.00  | 0.85     | 0.29     | 0.97   | 0.00  | 1.00 [1800]    |
| D     | 0.21   | 0.39  | 0.10     | 0.13     | 1.04   | 1.04  | 1.16 [1500]    |
| E     | 0.15   | 0.30  | -        | 0.12     | 1.00   | 1.00  | 1.00 [1050]    |

| Order 8 – Eff. ↑ (Is TT/TR format preserved? Yes or No) #Eva. ↓ |
| A     | 0.10   | 0.16  | -        | 0.03     | 0.88   | 0.48  | 1.00 [2850]    |
| B     | 0.09   | 0.43  | -        | -        | 0.61   | -     | 1.02 [2250]    |
| C     | 0.03   | 0.31  | -        | 0.02     | 1.16   | 0.49  | 1.11 [1320]    |
| D     | 0.20   | 0.53  | -        | -        | 1.03   | 0.32  | 1.06 [1950]    |
| E     | 0.33   | 0.33  | -        | -        | 1.17   | 0.23  | 0.88 [1500]    |

Figure 4. Average loss with varying the number of evaluations.

**Data generation.** We choose TT/TR, the most commonly used TN formats in machine learning, to generate tensor data. For each tensor order, i.e., the number of vertices, $N \in \{4, 6, 8\}$, we generate five tensors by randomly choosing ranks and values of vertices (core tensor). In more detail, the dimensions for each tensor modes are set to equal 3. Here we choose a small dimension as same as the one in (Li & Sun, 2020) because it is typically irrelevant to the searching difficulty, as shown in Theorem 3.1. Meanwhile, we uniformly select the TN-ranks from $\{1, 2, 3, 4\}$ in random, and i.i.d. draw samples from Gaussian distribution $N(0, 1)$ as the values of vertices. After contracting all vertices, we finally uniformly permute the tensor modes in random. The permutations maintain unknown for all algorithms in the experiment.

**Experiment setup.** In our method, we set the template $G_0$ as a cycle graph, the searching range for TN-ranks $R = 7$, the maximum iteration $\#Iter = 30$, the number of samples $\#Sample = 60$, and the tuning parameters $c_1 = 0.9$ and $c_2 = 0.9, 0.94, 0.98$ for three different tensor orders $N = 4, 6, 8$, respectively.

In the experiment, we also implement various TR decomposition methods with adaptive rank selection for comparison, including TR-SVD and TR-ALSAR (Zhao et al., 2016), Bayes-TR (Tao & Zhao, 2020), and TR-LM (Mickelin & Karaman, 2020). We also compare our method with two SO-TAs for TN-SS, including “Greedy” (Hashemizadeh et al., 2020) and TNGA (Li & Sun, 2020). Note that the original TNGA is forced to search only the TN formats. For a fair comparison, we extend it into two new versions: in “TNGA⁺” we trivially allow the method to search the formats and ranks simultaneously; and in “TNGA⁺⁺” we use the classic “random-key” trick (Bean, 1994) to encode $G_0$ into chromosomes, such that TNGA⁺ is capable of solving TN-PS as well but remaining the same genetic operations as TNGA. In these two methods and ours, we set the function
We now apply the proposed method to three benchmark problems on real-world data: image completion, image compression, and model compression of tensorial Gaussian process (TGP). In these problems, TT/TR has been widely exploited (Wang et al., 2017; Izmailov et al., 2018; Yuan et al., 2019a). The goal of the experiment is to verify whether imposing TN-PS can further improve their performance. Note that in all benchmarks TNGA+ and TNLS are implemented in the TR format. See Appendix for experimental details.

**Image completion.** TNLS is utilized to predict the missing values from natural images. In the experiment, seven images from USC-SIPI (Weber, 1997) are chosen, resized by $256 \times 256$, and then reshaped by visual data tensorization (VDT) (Latorre, 2005; Bengua et al., 2017; Yuan et al., 2019b) into a tensor of order-9. After that, the entries are uniformly removed at random with the missing rate $\{70\%, 90\%\}$, respectively. The average of the prediction RSE is shown in Table 4.

**Image compression.** Tensor decomposition methods are utilized to compress ten natural images randomly chosen from BSD500 (Arbelaez et al., 2010). In the data preparation phase, the images are grayscale, resized by $256 \times 256$, and then reshaped into order-8 tensors by both the VDT and the trivial reshaping operations as done in Python. The results are shown in Table 5, where we demonstrate the compression ratio (in the form of logarithm base 10) and the corresponding RSE. For comparison, we implement the methods, including TR-SVD, TR-LM, Greedy, TNGA+, and TNLS, in the experiment.
Table 5. Average compression ratio\(^*\) (log) and RSE\(^\dagger\) (in parentheses) for image compression, where “Reshape” and “VDT”, denote two different tensorization operations, and the values in square brackets are the number of evaluations required by methods.

|       | TR-SVD | TR-LM | Greedy | TNGA+     | TNLS       |
|-------|--------|-------|--------|-----------|------------|
| Reshape | 0.92 (0.15) | 0.90 (0.14) | 0.95 (0.15) | **1.36 (0.13)** [5700] | 1.32 (0.14) [1876] |
| VDT    | 1.10 (0.15) | 1.07 (0.15) | 0.91 (0.15) | 1.28 (0.15) [5700] | **1.30 (0.15)** [1546] |

Table 6. Number of parameters \((\times1000)\) and mean square error (MSE, in round brackets) for TGP model compression, where CCPP, MG and Protein are three datasets, and the values in [square brackets] show the number of evaluations required in each method.

|       | CCPP   | MG     | Protein |
|-------|--------|--------|---------|
| TGP   | 2.64 (0.06) [N/A] | 3.36 (0.33) [N/A] | 2.88 (0.74) [N/A] |
| TNGA+ | **2.24 (0.06)** [1500] | **3.01 (0.33)** [4900] | 2.03 (0.74) [3900] |
| TNLS  | 2.24 (0.06) [1051] | 3.01 (0.33) [3901] | **1.88 (0.74)** [3601] |

**Compressing TGP models.** In this task, we consider compressing not data but parameters of a learning model. To be specific, we compress the well-trained high-dimensional variational mean of TGP (Izmailov et al., 2018) by tensor decomposition. We evaluate the performance using three datasets for the regression task, including CCPP (Tufekci, 2014), MG (Flake & Lawrence, 2002), and Protein (Dua & Graff, 2017), for which we have the targeted tensors of the order-{4, 6, 9}, respectively. Table 6 shows the number of parameters \((\times1000)\) after decomposition and the corresponding mean square error (MSE, in the round brackets) for each dataset.

**Results.** We can observe from the experimental results that TN-PS can boost the performance of the TR models in all tasks. With the search of vertex permutations, i.e., the “mode-vertex” mappings, the expressive and generalization power of the TR models can be significantly improved. Compared with TN-SS methods like Greedy, TN-PS takes more “inductive bias” modeled as the template. As a consequence, imposing suitable “inductive bias” accelerates the searching process and helps avoid the loss in high-dimensional landscapes. Compared with TNGA+, TNLS achieves similar performance in three tasks and costs significantly less number of evaluations. This result is expected as we mentioned that the local sampling could leverage the more efficient “steepest-descent” path, which is not thought of in the GA-based methods.

6. Concluding Remarks

The experiential results demonstrate that TN-PS, a new variant of searching TN structures, can further improve the expressive power of TNs in various tasks. The new searching algorithm TNLS is verified as being more efficient than existing sampling-based algorithms, with fewer evaluations and faster convergence.

Our theoretical results analyze how the symmetry of TN formats determines the number of all possible “mode-vertex” mappings, i.e., the counting property, proving that a universal bound on the counting property exists if the TN formats are sufficiently sparse. We also establish the basic geometry of the search space for TN-PS. By the graph isomorphism relation of the TN structures, we construct a semi-metric function and prove its corresponding neighborhood for the search space. These results are applied as the theoretical foundation to the proposed sampling algorithm. Taken together, TN-PS explores more efficient TN structures for tensor decomposition/completion/learning tasks, preserving the TN formats in contrast to the previous TN-SS problem.

**Limitation.** One main limitation of our method is the higher running time compared with the greedy method (Hashemizadeh et al., 2020) in searching. A rigorous analysis about the smoothness of the landscape of TN-SS/PS also remains open. Our code is available at [https://github.com/ChaoLiAtRIKEN/TNLS](https://github.com/ChaoLiAtRIKEN/TNLS).

**Acknowledgements**

We appreciate the anonymous (meta-)reviewers of ICML 2022 for helpful comments. We are indebted to Chunmei for the help in proofreading the manuscript, Cichocki and Yokota for the insightful suggestions on the motivation, and Pan and Hashemizadeh for comments that improved the manuscript. This work was partially supported by JSPS KAKENHI (Grant No. 20K19875, 20H04249, 20H04208) and the National Natural Science Foundation of China (Grant No. 62006045, 62071132). Part of the computation was carried out at the Riken Alp Deep learning ENvironment (RAIDEN).

**References**

Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*,
Permutation Search of Tensor Network Structures via Local Sampling

Bachmayr, M., Schneider, R., and Uschmajew, A. Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations. *Foundations of Computational Mathematics*, 16(6):1423–1472, 2016.

Buck, T. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

Batselier, K. The trouble with tensor ring decompositions. *arXiv preprint arXiv:1811.03813*, 2018.

Bean, J. C. Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing*, 6(2):154–160, 1994.

Bengua, J. A., Phien, H. N., Tuan, H. D., and Do, M. N. Efficient tensor completion for color image and video recovery: Low-rank tensor train. *IEEE Transactions on Image Processing*, 26(5):2466–2479, 2017.

Cai, Y. and Li, P. A blind block term decomposition of high order tensors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6868–6876, 2021.

Hawkins, C. and Zhang, Z. Bayesian tensorized neural networks with automatic rank selection. *Neurocomputing*, 453:172–180, 2021.

Hayashi, K., Yamaguchi, T., Sugawara, Y., and Maeda, S.-i. Exploring unexplored tensor network decompositions for convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 5553–5563, 2019.

Huggins, W., Patil, P., Mitchell, B., Whaley, K. B., and Stoudenmire, E. M. Towards quantum machine learning with tensor networks. *Quantum Science and technology*, 4(2):024001, 2019.

Izmailov, P., Novikov, A., and Kropotov, D. Scalable Gaussian processes with billions of inducing inputs via tensor train decomposition. In *International Conference on Artificial Intelligence and Statistics*, pp. 726–735. PMLR, 2018.

Khavari, B. and Rabusseau, G. Lower and upper bounds on the pseudo-dimension of tensor network models. *Advances in Neural Information Processing Systems*, 34, 2021.
Permutation Search of Tensor Network Structures via Local Sampling

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kodryan, M., Kropotov, D., and Vetrov, D. Mars: Masked automatic ranks selection in tensor decompositions. *arXiv preprint arXiv:2006.10859*, 2020.

Kossaifi, J., Lipton, Z. C., Kolbeinsson, A., Khanna, A., Furlanello, T., and Anandkumar, A. Tensor regression networks. *Journal of Machine Learning Research*, 21:1–21, 2020.

Koutra, D., Parikh, A., Ramdas, A., and Xiang, J. Algorithms for graph similarity and subgraph matching. In *Proc. Ecol. Inference Conf*, volume 17, 2011.

Krasikov, I., Lev, A., and Thatte, B. Upper bounds on the automorphism group of a graph. *Discrete Math.*, 256 (math.CO/0609425):489–493, 2006.

Landsberg, J. M., Qi, Y., and Ye, K. On the geometry of tensor network states. *arXiv preprint arXiv:1105.4449*, 2011.

Latorre, J. I. Image compression and entanglement. *arXiv preprint quant-ph/0510031*, 2005.

Li, C. and Sun, Z. Evolutionary topology search for tensor network decomposition. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Li, C., Sun, Z., and Zhao, Q. High-order learning model via fractional tensor network decomposition. In *First Workshop on Quantum Tensor Networks in Machine Learning, 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

Li, N., Pan, Y., Chen, Y., Ding, Z., Zhao, D., and Xu, Z. Heuristic rank selection with progressively searching tensor ring network. *Complex & Intelligent Systems*, pp. 1–15, 2021.

Liu, J., Li, S., Zhang, J., and Zhang, P. Tensor networks for unsupervised machine learning. *arXiv preprint arXiv:2106.12974*, 2021.

Liu, S., Chen, P.-Y., Kailkhura, B., Zhang, G., Hero III, A. O., and Varshney, P. K. A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. *IEEE Signal Processing Magazine*, 37(5):43–54, 2020a.

Liu, S., Lu, S., Chen, X., Feng, Y., Xu, K., Al-Dujaili, A., Hong, M., and O’Reilly, U.-M. Min-max optimization without gradients: Convergence and applications to black-box evasion and poisoning attacks. In *International Conference on Machine Learning*, pp. 6282–6293. PMLR, 2020b.

Long, Z., Zhu, C., Liu, J., and Liu, Y. Bayesian low rank tensor ring for image recovery. *IEEE Transactions on Image Processing*, 30:3568–3580, 2021.

Lück, W. Survey on geometric group theory. *arXiv preprint arXiv:0806.3771*, 2008.

Markov, I. L. and Shi, Y. Simulating quantum computation by contracting tensor networks. *SIAM Journal on Computing*, 38(3):963–981, 2008.

Mickelin, O. and Karaman, S. On algorithms for and computing with the tensor ring decomposition. *Numerical Linear Algebra with Applications*, 27(3):e2289, 2020.

Miller, J., Rabusseau, G., and Terilla, J. Tensor networks for probabilistic sequence modeling. In *International Conference on Artificial Intelligence and Statistics*, pp. 3079–3087. PMLR, 2021.

Nie, C., Wang, H., and Tian, L. Adaptive tensor networks decomposition. In *BMVC*, 2021.

Novikov, A., Podoprikhin, D., Osokin, A., and Vetrov, D. P. Tensorizing neural networks. In *Advances in Neural Information Processing Systems*, pp. 442–450, 2015.

Novikov, G. S., Panov, M. E., and Oseledets, I. V. Tensor-train density estimation. In *Uncertainty in Artificial Intelligence*, pp. 1321–1331. PMLR, 2021.

Oseledets, I. V. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

Papadimitriou, C. H. and Yannakakis, M. The complexity of facets (and some facets of complexity). In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pp. 255–260, 1982.

Qiu, H., Li, C., Weng, Y., Sun, Z., He, X., and Zhao, Q. On the memory mechanism of tensor-power recurrent models. In *International Conference on Artificial Intelligence and Statistics*, pp. 3682–3690. PMLR, 2021.

Razin, N., Maman, A., and Cohen, N. Implicit regularization in tensor factorization. In *International Conference on Machine Learning*, pp. 8913–8924. PMLR, 2021.

Razin, N., Maman, A., and Cohen, N. Implicit regularization in hierarchical tensor factorization and deep convolutional neural networks. *arXiv preprint arXiv:2201.11729*, 2022.

Reyes, J. and Stoudenmire, M. A multi-scale tensor network architecture for classification and regression. *arXiv preprint arXiv:2001.08286*, 2020.
Richter, L., Sallandt, L., and Nüsken, N. Solving high-dimensional parabolic PDEs using the tensor train format. In Meila, M. and Zhang, T. (eds.), Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pp. 8998–9009. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/richter21a.html.

Russell, S. and Norvig, P. Artificial Intelligence: A Modern Approach. CUMINCAD, 1995.

Savarese, P., McAlester, D., Babu, S., and Maire, M. Domain-independent dominance of adaptive methods. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16286–16295, 2021.

Sedighin, F., Cichocki, A., and Phan, A.-H. Adaptive rank selection for tensor ring decomposition. IEEE Journal of Selected Topics in Signal Processing, 15(3):454–463, 2021.

Singh, S. Continuum-armed bandits: A function space perspective. In International Conference on Artificial Intelligence and Statistics, pp. 2620–2628. PMLR, 2021.

Snyder, L. V. and Daskin, M. S. A random-key genetic algorithm for the generalized traveling salesman problem. European journal of operational research, 174(1):38–53, 2006.

Tao, Z. and Zhao, Q. Bayesian tensor ring decomposition for low rank tensor completion. In International Workshop on Tensor Network Representations in Machine Learning, IJCAI, 2020.

Tüfekci, P. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. International Journal of Electrical Power & Energy Systems, 60:126–140, 2014.

Verstraete, F. and Cirac, J. I. Renormalization algorithms for quantum-many body systems in two and higher dimensions. arXiv preprint cond-mat/0407066, 2004.

Wang, W., Aggarwal, V., and Aeron, S. Efficient low rank tensor ring completion. In Proceedings of the IEEE International Conference on Computer Vision, pp. 5697–5705, 2017.

Weber, A. G. The USC-SIPI image database version 5. USC-SIPI Report, 315(1), 1997.

Ye, K. and Lim, L.-H. Tensor network ranks. arXiv preprint arXiv:1801.02662, 2019.

Yokota, T., Zhao, Q., and Cichocki, A. Smooth parafac decomposition for tensor completion. IEEE Transactions on Signal Processing, 64(20):5423–5436, 2016.
In the appendix, we first give the proofs for the results mentioned in the main body of the paper. After that, more details of the experiments, including tuning parameter settings and additional experimental results, will be introduced.

A. Proofs

A.1. Proof of Lemma 3.2

Proof. To obtain the result, we first have the following inequalities:

\[
\log |\text{Aut}(G_0)| - \log |S_N| \leq \log N + \log \Delta! + (N - \Delta - 1) \log(\Delta - 1) - \log N!
\]
\[
\leq (1/2 - N) \log N + (\Delta + 1/2) \log \Delta + (N - \Delta - 1) \log(\Delta - 1) + N - \Delta - \frac{1}{12N + 1} + \frac{1}{12\Delta}
\]
\[
\leq (1/2 - N) \log N + N - \Delta + 1 = (1/2 - N) \log d + N - N/d + 1/12,
\]

where the first inequality follows from Theorem 2 given in (Krasikov et al., 2006). In this theorem, it is proved that \( |\text{Aut}(G_0)| \) is above bounded by the maximum graph degree, written \( \Delta \), as follows:

\[
\log |\text{Aut}(G_0)| \leq \log N + \log \Delta! + (N - \Delta - 1) \log(\Delta - 1).
\]

The second inequality of (10) follows by \( |S_N| = N! \) and Stirling approximation of factorials, by which the terms \( \log \Delta! \) and \( -\log N! \) are bounded as follows:

\[
\log \Delta! \leq 0.5 \log 2\pi + (\Delta + 1/2) \log \Delta - \Delta + \frac{1}{12\Delta},
\]

and

\[
-\log N! \leq -0.5 \log 2\pi - (N + 1/2) \log N + N - \frac{1}{12N + 1},
\]

respectively. In the third line of (10), the (in-)equalities follows from: \( \log(\Delta - 1) \leq \log(\Delta) \), \( 1/(12\Delta) \leq 1/24 \) and \( N > 0 \), and the assumption \( N/\Delta = d \). The proof is thus accomplished by eliminating the logarithm on the both sides of the inequality.

A.2. Proof of Theorem 3.1

Proof. According to the Lagrange’s theorem in group theory, the size of \( L_{G_0, R} \) is equal to

\[
|L_{G_0, R}| = \frac{|S_N| \cdot |F_{G_0, R}|}{|\text{Aut}(G_0)|} = \frac{|S_N| \cdot |Z_R|^{\|E_0\|}}{|\text{Aut}(G_0)|},
\]

where the equation \( |F_{G_0, R}| = |Z_R|^{\|E_0\|} \) holds by the TN-PS model. By the handshaking lemma in graph theory,

\[
|E_0| = \frac{1}{2} \sum_{n=1}^{N} \text{deg}(v_n),
\]

where \( G_0 = (V, E_0), v_n \in V \) for \( n \in [N] \), and \( \text{deg}(v_n) \) denotes the degree of \( v_n \). The number of edges is thus bounded by

\[
\frac{N}{2} \delta \leq |E_0| \leq \frac{N}{2} \Delta.
\]

The inequalities (5) in Theorem 3.1 are finally obtained by combing Lemma 3.2 to (14) and (16).

A.3. Proof of Lemma 3.3

Proof. First, we prove that \((G_0, d_{G_0})\) defines a semi-metric space. To do this, we should prove the function \( d_{G_0} \) defined by (7) satisfying the following claims:
Thus, there exist the proof of the lemma is accomplished.

for $G_1, G_2, G \in \mathbb{G}_0$. We first see that the three claims are naturally true for a trivial $\mathbb{G}_0$. Then in the following we only consider the case of non-trivial $\mathbb{G}_0$. To prove the claim (a), we suppose $G_1 \neq G_2$ with $G_1 = g_1 \cdot G_0, G_2 = g_2 \cdot G_0$. It thus give $g_1 \neq g_2$ holds. $g_i \cdot \text{Aut}(G_0) \cap g_j \cdot \text{Aut}(G_0) = 0$, since $g_i \cdot \text{Aut}(G_0), i = 1, 2$ are left cosets of $\text{Aut}(G_0)$ which partitions $\mathbb{S}_N$. We therefore have $p_1 \neq p_2$ and $d_T(p_1, p_2) > 0, \forall p_i \in g_i \cdot \text{Aut}(G_0), i = 1, 2$, by which $d_{\mathbb{G}_0}(G_1, G_2) > 0$. Suppose conversely that $G_1 = G_2$, then we have $g_1 \cdot g_2 \in \text{Aut}(G_0)$. We thus know that there exist $p \in g_1 \cdot \text{Aut}(G_0)$ such that $p = g_1 g_2 = g_2$. Therefore, it obeys

$$d_{\mathbb{G}_0}(G_1, G_2) = \min_{p_i \in g_i \cdot \text{Aut}(G_0), i = 1, 2} d_{\mathbb{T}}(p_1, p_2) = d_{\mathbb{T}}(p_1, p_2) = d_{\mathbb{T}}(G_1, G_2) = 0.$$  \hspace{1cm} (17)

By $d_{\mathbb{G}_0}(G_1, G_2) \geq 0$, the claim (a) is thus proved.

The claim (b) is obviously true.

Next, we show the set $\mathbb{N}_N(G)$ defines the neighborhood of $G$ in $\mathbb{G}_0$ associated with $d_{\mathbb{G}_0}$. We first see that it is trivially true if $D = 0$. For $D > 0$, we prove that $d_{\mathbb{G}_0}(G', G) \leq D$ holds for all $G' \in \mathbb{N}_D(G)$. By the assumption $G = g \cdot G_0$ we first have that $p \cdot G_0 = gA \cdot G_0 = g \cdot G_0$ holds for all $p \in g \cdot \text{Aut}(G_0)$ where $A \in \text{Aut}(G_0)$. By $G' \in \mathbb{N}_D(G)$, there exists $d_0 \in [D]$ such that $G' \in \mathbb{N}_{d_0}(G)$. Thus

$$d_{\mathbb{G}_0}(G', G) = \min_{p' \in g' \cdot \text{Aut}(G_0)} d_{\mathbb{G}}(p', p)$$

$$\leq d_{\mathbb{T}} \left( gA_1 \prod_{i=1}^{d_0} t_i A_2, gA_3 \right)$$

$$= d_{\mathbb{T}} \left( 1, \left( gA_1 \prod_{i=1}^{d_0} t_i A_2 \right)^{-1} gA_3 \right)$$

$$= d_{\mathbb{T}} \left( 1, A_2^{-1} \prod_{i=1}^{d_0} t_i \right) d_{\mathbb{T}} \left( 1, \prod_{i=1}^{d_0} t_{d_0-i} \right)$$

where $A_i \in \text{Aut}(G_0), i = 1, 2, 3, A_1 = A_3$ and $A_2$ is equal to the identity of $\text{Aut}(G_0)$. In (18), the fist line follows from the definition of $\mathbb{L}_d(G)$; the third line holds the left-isometry property of the world metric; the last line holds by the definition of the word metric and the fact $t_i^{-1} = t_i \in \mathbb{T}$. Next, we prove the converse side, that is, $G_x \in \mathbb{N}_D(G)$ for all $G_x \in \{ G' \in \mathbb{G}_0 | d_{\mathbb{G}_0}(G', G) \leq D \}$. By the definition of $d_{\mathbb{G}_0}$

$$d_{\mathbb{G}_0}(G_x, G) = \min_{p_x \in g_x \cdot \text{Aut}(G_0); p \in g \cdot \text{Aut}(G_0)} d_{\mathbb{T}}(p_x, p) \leq D.$$  \hspace{1cm} (19)

Thus, there exist $A_x, A \in \text{Aut}(G_0)$ such that the inequality

$$d_{\mathbb{T}}(g_x A_x, pA) = d_{\mathbb{T}}(1, A_x^{-1} g_x^{-1} gA) \leq D$$  \hspace{1cm} (20)

holds. Let $g_x = g \cdot h_x$ for some $h_x \in \mathbb{S}_N$, then $d_{\mathbb{T}}(1, A_x^{-1} h_x^{-1} A) \leq D$. According to the definition of the word metric $d_{\mathbb{T}}$, we know there exists $d \leq D$ and a sequence of permutations $\{ t_1, t_2, \ldots, t_d \} \subset \mathbb{T}_N$ such that $A_x^{-1} h_x^{-1} A = \prod_{i=1}^{d} t_i$, where $\epsilon = \{ +1, -1 \}$ (Lück, 2008). Since $t_i^{-1} = t_i$, the equation $h_x = A \prod_{i=1}^{d} t_i A_x^{-1}$ holds, and $G_x = g_x G_0 = gh_x G_0 = gA \prod_{i=1}^{d} t_i A_x^{-1} G_0 = gA \prod_{i=1}^{d} t_i G_0$ consequently holds, where the last equation follows from $A_x^{-1} \in \text{Aut}(G_0)$. The equations say that $G_x$ is an element of $\mathbb{L}_d(G)$, namely, $G_x \in \mathbb{L}_d(G) \subseteq \mathbb{N}_D(G)$. Combing the results from the both two sides, the proof of the lemma is accomplished.
A.4. Proof of Theorem 3.4

Before the proof of Theorem 3.4, we first restate a classic claim in group theory about the cycle decomposition of a permutation under conjugation.

**Lemma A.1.** Let $\sigma$ and $\tau$ be two elements of $S_N$. Suppose that $\sigma = (a_1, a_2, \ldots, a_k)(b_1, a_2, \ldots, b_l)$ is the cycle decomposition of $\sigma$. The $\tau\sigma\tau^{-1}$, the conjugate of $\sigma$ by $\tau$, is $\tau(a_1), \tau(a_2), \ldots, \tau(a_k)(\tau(b_1), \tau(a_2), \ldots, \tau(b_l))$ is the cycle decomposition of $\tau\sigma\tau^{-1}$.

We then apply this lemma to the following result:

**Lemma A.2.** Let $C_N = \{(i, j) | 1 \leq i < j \leq N\}$ be the collection of all 2-cycles of $S_N$. For any $G' \in \mathbb{I}_d(G)$ where $G = g \cdot G_0 \in G_0$, there exist a series of 2-cycles, i.e., $c_1, c_2, \ldots, c_d \in C_N$ such that $G' = \prod_{k=1}^d c_k G$.

**Proof.** We first consider the case $d = 1$. By $G' \in \mathbb{I}_d(G)$, there exist $A' \in Aut(G_0)$ and $t \in T_N$ such that $G' = gA'tG_0$. Let $c \in S_N$ be a permutation satisfying $G' = c \cdot G$, then we know $G' = c \cdot G = c_1gA \cdot G_0$ for any $A \in Aut(G_0)$. Combining the above equations,

$$c = gA't(gA')^{-1} \quad (21)$$

by $A = A'$, implying that $c$ is the conjugate of $t$ by $gA'$. Applying Lemma A.1 to (21), we have that $c \in C_N$ is true. Next, we extend it to the case $d > 2$. In this we have the equation $G' = gA't_1t_2G_0$ holding for $t_1, t_2 \in T_N$. We further assume that $c_1, c_2 \in S_N$ and $c_1 = (gA')t_1(gA')^{-1}$, such that $G' = c_1c_2G$. So we can have the following equations:

$$G' = gA't_1t_2 \cdot G_0 = c_1gA't_2 \cdot G_0 = c_1gA'(gA')^{-1}gA'G_0 = c_1gA'(gA')^{-1}G,$$

where the last equation holds for $G = gA'G_0$. We thus have $c_2 = gA't_2(gA')^{-1}$, namely, the conjugate of $t_2$ by $gA'$. Applying Lemma A.1 to these equations, we know that $c_1, c_2 \in C_N$. Lemma A.2 is proved by extending the above procedure to the cases $d > 2$.

Last, to prove Theorem 3.4, we see that swapping two vertices using Alg. 1 is equivalent to acting a 2-cycle from $C_N$ on the vertices of the graph. Since for all $i_k, j_k \in [N], i_k \neq j_k \in [d]$ can be sampled with a positive probability, it deduces that any two-cycles $c_k = (i_k, j_k) \in C_N$ can be drawn with a positive probability using Alg. 1, covering $\mathbb{I}_d(G)$ according to Lemma A.2.

B. Experiment details and additional results

B.1. TNGA+: an Extension of TNGA for TN-PS

In this subsection, we briefly explain how TNGA+, an extension of TNGA (Li & Sun, 2020) for TN-PS, encodes the vertex permutations into chromosomes, which are used to seek for the optimal TN structures by genetic operators.

Figure 5 depicts the encoding process. We encode the structures for TN-PS from two ingredients, the TN-ranks and the permutations, respectively. For the former, by $F_{G_0} = \mathbb{Z}^{+ \cdot |E_0|}$, the ranks can be directly encoded into a string of dimension $|E_0|$ with their coordinates in $\mathbb{Z}^{+ \cdot |E_0|}$.

For the latter, we randomly embed a permutation into the space $[0, 1]^{|V|}$, a set of decimal number vectors, by a random-key trick (Bean, 1994), which is popularly used to solve the optimal sequencing tasks. More precisely, the random-key representation encode a permutation with a vector of random numbers from $[0, 1]$, and the order of these random numbers reflects the permutation. For instance, the code $(0.46, 0.91, 0.33)$ would represent the permutation $2 \rightarrow 3 \rightarrow 1$. Finally, the encoded strings are simply the concatenation of the two ingredients.

B.2. Synthetic Data in TT/TR Format

**Configuration of TNGA**, TNGA+. Both of these two algorithms are based on the GA framework (Li & Sun, 2020). Throughout the TT/TR format synthetic data experiments, they share the same parameters listed as follows. The maximum number of generations is set to be 30. The population in each generation is set to be 150 under all settings. During each generation in GA, the elimination rate is 36%. The reproduction trick in (Snyder & Daskin, 2006) is adopted and we set the reproduction number to be 2. Meanwhile, for the selection probability of the recombination operation, we set hyper-parameters $\alpha = 20$ and $\beta = 1$. Moreover, there is a chance of 24% for each gene to mutate after the recombination.
finished. We initialize the vertices (core tensors) with each element \( i.i.d. \) sampled from Gaussian distribution \( N(0, 0.1) \). We set the learning rate of the Adam optimizer (Kingma & Ba, 2014) to 0.001. The decomposition for each individual is repeated 4 times.

**Experiment setup of Figure 4 in the manuscript.** In this experiment, the order-8 tensor is selected from the TR structure search experiment. For the order-12 data, we uniformly choose TR-ranks from \( \{1, 2, 3, 4\} \) and set the dimension of all tensor modes to 3. The values of vertices are drawn from Gaussian distribution \( N(0, 1) \). After contracting all vertices, we finally uniformly permute the tensor modes in random. For TNGA+ and TNLS, all the parameters are set the same as in the TR structure search experiment, except that the population of TNGA+ and the sample number of TNLS are set to be 60 or 100.

**Trade off between model complexity and approximation accuracy.** In the experiment, the tuning parameter \( \lambda \) given in (9) balances the influence of model complexity and approximation accuracy in the searching process. Figure 6 shows how RSE and \( \text{Eff.} \) values change with the varying of \( \lambda \). In more details, we choose the values of \( \lambda \) from \( \{0.1, 1, 10, 100, 1000\} \) and calculate the RSE and \( \text{Eff.} \) averaged over the data used in the synthetic TT/TR data experiments of the order \( \{4, 6, 8\} \), respectively. Other experiment configuration remains the same as used in the experiment. We can see from Figure 6 that the \( \text{Eff.} \) values are larger than 1 consistently with a wide range of \( \lambda \) in all the three orders. It implies that the TNLS method is relatively stable with the varying of the parameter \( \lambda \) if the tensor is generated with TN models. The result is expected since in this case the RSE will decrease dramatically once a good TN structure is found, so the value \( \lambda \cdot \text{RSE} \), the second term of the objective function in (9), is neglected compared with the first term corresponding to the model complexity. However, note that the stability would be not held if the tensor is not in low-rank TN formats such as those tensorized natural images.

**B.3. Synthetic Data in Other TN Format**

**Data Generation.** For the synthetic data generation of TTree (order-7) (Ye & Lim, 2019), PEPS (order-6) (Verstraete & Cirac, 2004), hierarchical Tucker (HT, order-6) (Hackbusch & Kühn, 2009) and multi-scale entanglement renormalization ansatz (MERA, order-8) (Cincio et al., 2008; Reyes & Stoudenmire, 2020) which the structures are demonstrated in Figure 7, we first set the dimensions of each tensor mode to 3 and uniformly randomly generate the TN-ranks from \( \{1, 2, 3, 4\} \). Then, each element of the cores is generated \( i.i.d. \) from Gaussian distribution \( N(0, 0.1) \). After contracting all vertices, we finally uniformly permute the tensor modes in random.
Permutation Search of Tensor Network Structures via Local Sampling

Figure 7. Illustration of the TN structures applied in the synthetic experiment. The blue nodes with an outer indices indicate the external cores and the orange nodes indicate the internal cores.

Figure 8. Illustration of the employed images in image completion experiment.

Configuration of the comparing methods. For TNGA+ and TNLS, the parameters are set as same as the TR structure search experiment, except that for TNGA+ the population in each generation is increased to be 120. Moreover, the coding schemes for HT and MERA are different from TTree and PEPS, which only contain external cores (vertices of color blue). Specifically, for HT and MERA, we fix the permutation of the internal cores (vertices of color orange), and therefore only encode the permutation of the external cores. The experimental results including the evaluation numbers of TNGA+ and TNLS are shown in Table 3.

B.4. Real-World Data

Image completion. In this experiment, we consider uniformly random missing with the missing rates 70% and 90%. In specific, we firstly use Matlab command “randperm” to generate random integer sequence with length that equals to the number of image elements. Then, according to the missing rate, we select a subset of this sequence to generate a binary mask tensor with the same size as the image. Finally, using this mask, we can generate the missing image. For recovery performance evaluation, we use the RSE of predicted values on the missing entries.

For the proposed TNLS, we set the the maximum iteration #Iter = 30, and tuning parameters $c_1 = 0.95$, $c_2 = 0.9$, and the number of sampling #Sample = 150. Moreover, the rank bound, the learning rate of Adam, and the variance of the Gaussian distribution for core tensors initialization are set to 14, 0.001, 0.1 respectively. For the trade-off parameter $\lambda$, we set it as 0.0008, 0.0007 for missing rate 0.7, 0.9. For TNGA+, the maximum number of the generations is set to be 30. The elimination rate is 10% and the reproduction number is set to be 1. Moreover, we set $\alpha = 20$ and $\beta = 1$. The chance for each gene to mutate after the recombination is 24%. Other settings, including $\lambda$, Gaussian distribution for initialization, the rank bound and the learning rate, are the same with TNLS.

Image compression. In the experiment, we randomly select 10 natural images from the BSD500 (Arbelaez et al., 2010)$^1$

$^1$https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/BSDS300/html/dataset/images.html
The images used in this experiment are shown in Figure 9. We use the Matlab commands “resize” and “rgb2gray” to turn them into gray-scaled images of size $256 \times 256$, and then rescale them to $[0, 1]$. Moreover, in this section, we tensorized these images into order-8 tensors by two different ways: a directly reshaping operation denoted by “Reshape” and visual data tensorization (Latorre, 2005; Bengua et al., 2017; Yuan et al., 2019b), a image-resolution-based tensorization method, denoted by “VDT”.

For the proposed TNLS, we set the the maximum iteration $\# \text{Iter} = 20$, and tuning parameters $c_1 = 0.95$, $c_2 = 0.9$, and the number of sampling $\# \text{Sample} = 150$. Moreover, the rank bound, the learning rate of Adam, and the variance of the Gaussian distribution for core tensors initialization are set to be $14$, $0.01$, $0.1$, respectively. For the trade-off parameter $\lambda$, we set it as $5$. For TNGA+, the maximum number of the generations is set to be $30$. The population in each generation are set to be $300$. The the elimination rate is $10\%$ and the reproduction number is set to $1$. Moreover, we set $\alpha = 25$ and $\beta = 1$. The chance for each gene to mutate after the recombination is $30\%$. Other settings, including $\lambda$, Gaussian distribution for initialization, the rank bound and the learning rate, are the same with TNLS.

**Compressing TGP models.** In this task, we choose three univariate regression datasets from the UCI and LIBSVM archives. The Combined Cycle Power Plant (CCPP) dataset consists of 9569 data points collected from a power plant with 4 features and a single response. The MG data have 1385 data points with 6 features and a single response. The Protein data contain 45730 instances with 9 attributes and a single response. For all the datasets, we randomly choose $80\%$ of the data for training and the rest for testing, then standardize the training and testing sets respectively by removing the mean and scaling to unit variance, which is the same with settings in TTGP (Izmailov et al., 2018).

In this experiment, we aim to demonstrate that our method is capable of searching more efficient structures in this learning task. This is different from the above tasks since we search for TN structures of model parameters, instead of compressing data. Specifically, tensor train Gaussian process (TTGP) tensorizes the variational mean vector in GP to a tensor whose order equals to the number of input features, and the dimension of each order is the number of inducing points on the corresponding feature. In our settings, for CCPP, we choose 12 inducing points on each feature and result in an order-4 tensor of shape $12 \times 12 \times 12 \times 12$. For MG, we choose 8 inducing points and get an order-6 tensor of shape $8 \times 8 \times 8 \times 8 \times 8 \times 8$. For Protein, we choose 4 inducing points and obtain an order-9 tensor of shape $4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 4$. TTGP uses TT to approximate these tensors. However, the original TTGP are restricted to TT representation and the TT-ranks are treated as pre-defined hyper-parameters. For all the datasets, we set TT-ranks as $10$.

To learn more compact representations, we apply the structure searching to TTGP. In particular, we firstly train a TTGP with given TT-ranks and get the TT representation of the variational mean. Then we use our method to search for alternative TR structures of the TT variational mean. Finally, we plug the reparameterized variational mean back into the original TTGP model for inference. In summary, we follow the settings of TTGP except that we reparameterize the TT tensor.

For the proposed TNLS, we set the the maximum iteration $\# \text{Iter} = 20$, and tuning parameters $c_1 = 0.9$, $c_2 = 0.9$, and the number of sampling $\# \text{Sample} = 150, 300, 300$ for the TT variational mean of CCPP, MG and Protein regression

---

1. https://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant
2. https://archive.ics.uci.edu/ml/datasets/Physicochemical+Properties+of+Protein+Tertiary+Structure
3. https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html#mg
4. https://archive.ics.uci.edu/ml/datasets/Physicochemical+Properties+of+Protein+Tertiary+Structure
task. Moreover, the rank bound, the learning rate of Adam, and the variance of the Gaussian distribution for core tensors initialization are set to be $14, 0.001, 0.01$, respectively. For the trade-off parameter $\lambda$, we set it as $\lambda = 1 \times 10^5, 1 \times 10^7, 1 \times 10^3$ for CCPP, MG and Protein, respectively. For TNGA+, the maximum number of the generations is set to be 30. The population in each generation are set to be 150, 190, 300 for the TT variational mean of CCPP, MG and Protein regression task. The elimination rate is 30% and the reproduction number is set to 1. Moreover, we set $\alpha = 20$ and $\beta = 1$. The chance for each gene to mutate after the recombination is 30%. Other settings, including $\lambda$, Gaussian distribution for initialization, the rank bound and the learning rate, are the same with TNLS.