Evolutionary Topology Search for Tensor Network Decomposition

Chao Li[*1] Zhun Sun[*1]

Abstract

Tensor network (TN) decomposition is a promising framework to represent extremely high-dimensional problems with few parameters. However, it is challenging to search the (near-)optimal topological structures for TN decomposition, since the number of candidate solutions exponentially grows with increasing the order of a tensor. In this paper, we claim that the issue can be practically tackled by evolutionary algorithms in an affordable manner. We encode the complex topological structures into binary strings, and develop a simple genetic meta-algorithm to search the optimal topology on Hamming space. The experimental results by both synthetic and real-world data demonstrate that our method can effectively discover the ground-truth topology or even better structures with a small number of generations, and significantly boost the representational power of TN decomposition compared with well-known tensor-train (TT) or tensor-ring (TR) models. Our code is available at `https://github.com/minogame/icml2020-TNGA`.

1. Introduction

How to use fewer parameters to represent data and models is a crucial issue in both machine learning and scientific computing (Bellman, 1966; Bishop, 2006; Cichocki, 2014). To tackle the problem, tensor decomposition (TD), including CANDECOMP/PARAFAC decomposition (Bro, 1997), Tucker decomposition (Tucker, 1966) and their variants (De Lathauwer, 2008), becomes a promising framework, which decomposes high-order tensors into low-dimensional forms (Kolda & Bader, 2009). In real-world applications, TD has been successfully applied to model compression (Bahadori et al., 2014), multi-modality and multi-task learning (Zadeh et al., 2017; Liu et al., 2018), latent variable model (Anandkumar et al., 2014), data restoration (Zhang et al., 2019) to name a few.

More recently, some studies brought the idea from physics to extend TD into a more sophisticated form, *i.e.*, tensor network (TN) (Cichocki, 2014; Cichocki et al., 2016; Khoromskij, 2018), which derives various models such as tensor train (TT) (Oseledets, 2011), tensor ring (TR) (Zhao et al., 2016), hierarchical Tucker (HT, or tensor tree) (Grasedyck, 2010) and their variants (Phan et al., 2018; Huang et al., 2019). More interestingly, lots of studies in machine learning reported the overwhelming performance of the TN-based methods compared to the conventional TD's (Novikov et al., 2015; Stoudenmire & Schwab, 2016; Wang et al., 2017; Levine et al., 2017; Wang et al., 2018; Cheng et al., 2019; Hayashi et al., 2019). While on the other hand, we notice that the existing TN-based methods are mainly developed by specifying the *topology* of the model, such as line, cycle, tree and grid (*a.k.a.*, multiple cycles). Such a fact raises the question *whether there exist "better or even optimal" topological structures of TN decomposition for a given tensor, and how to efficiently search them?* It is naturally expected that the optimal topology of TN would be out of the box of the manucrafted structures yet with more promising representational power.

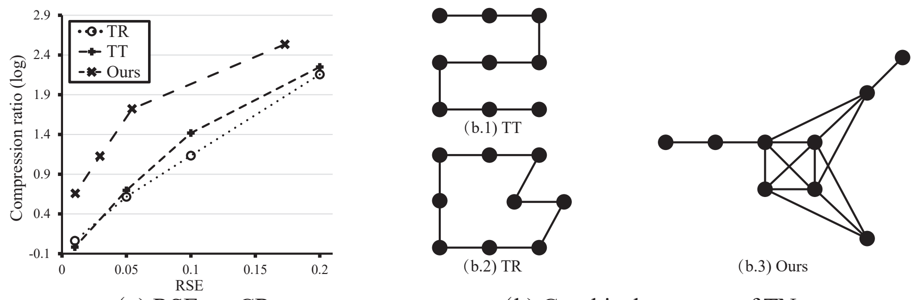
However, it is an open problem to search the optimal topology and the challenges are mainly twofold: Theoretically, this problem seems NP-hard[1] in the optimization context; Practically, the solution space is unacceptably huge even for a small-scale problem. For instance, there are more than 68 billion candidate solutions for only an order-9 tensor (known by Proposition 2). As related works, although there are some studies focusing on rank (*a.k.a.* bound dimension in physics) estimation (Grasedyck, 2010; Oseledets, 2011; Zhao et al., 2016; Yokota et al., 2018; Mickelin & Karaman, 2018), they generally assume to fix the topology. To our best knowledge, there is few discussion on the topology search issue so far.

In this paper, we address the topology search issue *practically*. The core idea is to formulate the problem as a

[*]Equal contribution [1]Center for Advanced Intelligence Project, RIKEN Institue, Tokyo, Japan. Correspondence to: Chao Li <chao.li@riken.jp>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

[1]The rigorous proof of this claim still seems to be unsolved. The authors infer this conjecture based on the empirical knowledge and the fact "Most tensor problems are NP-hard (Hillar & Lim, 2013).".



(a) RSE *vs.* CR (b) Graphical strctures of TN

Figure 1. An illustrative example to approximate a tensorized "lena" image (order-9 tensor) by TN decomposition. In the experiment, we apply GA to searching the topological structures from the data without careful "rank" tuning. For comparison, we also decompose the data by TT-SVD (Oseledets, 2011) and TR-SVD (Zhao et al., 2016), by which the rank of the model are automatically determined. Plot (a) illustrates the compression ratio achieved by different structures under various relative square errors (RSE), and plot (b) shows the topological structures used in the experiment.

combinatorial optimization on Hamming space, and to apply the well-developed genetic algorithm (GA) to searching the solution. Concretely, we first reforge the existing graphical representation (GR) of TN (Cichocki et al., 2016; Ye & Lim, 2019), such that there exists a bijection between the TNs' topology and a simple graph. Subsequently, we encode each graph into a fixed-length binary string by its adjacency matrix. Taking the encoded binary strings as chromosomes, we can apply various GAs to searching the topological structures from the tensor. Figure 1 shows a demo experiment, in which we employ TT-SVD (Oseledets, 2011), TR-SVD (Zhao et al., 2016) and our method to approximate a tensorized "lena" image (an order-9 tensor) under a series of relative square errors (RSE) conditions. It is clear that the new discovered structure (b.3) results in significantly less number of parameters to represent the data than TT and TR under all RSE conditions. The details about the experimental settings are given in Section 4.3.

1.1. Related Works

Tensor network (TN). The studies on TN have long history in physics (Fannes et al., 1992; Hübener et al., 2010; Calabrese et al., 2011; Orús, 2014), in which the topological structures of TN models are generally assumed to be known in advance. On the other side, in the communities of scientific computing and machine learning, TN is applied as a non-linear approximation of high-dimensional problems (Kolda & Bader, 2009; Sidiropoulos et al., 2017), and up until now only simple topological structures are studied such as line (TT (Oseledets, 2011)), cycle (TR (Zhao et al., 2016)), tree, (HT (Grasedyck, 2010)) and their combinations (Phan et al., 2018; Huang et al., 2019). As to the model selection issue in TN, the existing studies mainly focus on "TN rank determination" (Grasedyck, 2010; Yokota et al., 2018; Mickelin & Karaman, 2018). In contrast, the focus of our work is on providing an approach to search better *topological* structures of TN models.

Genetic algorithm (GA). As a population-based evolutionary method, GA has been paid much attention since the last century (Whitley, 1994; Harik et al., 1999; Maulik & Bandyopadhyay, 2000) to new a few. The recent boost on studies of GA is due to the interest on neural network architecture search (NAS) (Irani & Nasimi, 2011; Elsken et al., 2018; Lu et al., 2019; Liang et al., 2018; Elsken et al., 2019; Real et al., 2019). Inspired by the recent studies, we also consider GA as a promising algorithm in this paper, yet the focus of our work is mainly on *building a bridge from topology search problem of TN decomposition to GA,* which is entirely different problem from NAS or designing new approaches for GA.

2. Graphical Representation of TN

To establish a bijective mapping from TN's topology to the simple graph (in sense of graph theory), we reforge the existing definition of TN graphical representation. After that, we propose a formal statement on the topology search issue of TN decomposition.

2.1. Tensor Network (TN)

First, we briefly recall the definitions and notations about tensor and tensor network (TN) for the self-contained consideration. Specific details about TN and its extension are given in (Ye & Lim, 2019) and the references therein.

Throughout the paper, we define a *tensor* as a multi-dimensional array of real numbers (Kolda & Bader, 2009). The order of a tensor is defined as the number of indices. Thus, an order-0 tensor is a scalar denoted by italic lowercase letters, *e.g.* $x \in \mathbb{R}$, an order-1 tensor is a vector denoted by boldface lowercase letter, *e.g.* $\mathbf{a}, \mathbf{b} \in \mathbb{R}^I$, and an order-2 tensor is a matrix denoted by boldface capital letter, *e.g.*

$\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$. For higher-order tensors, we denote them by calligraphic letters, *e.g.* $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$. Sometimes we also use the calligraphic letters to represent vectors or matrices without ambiguity.

As an extension of matrix product, an *index contraction* (also called contracted product of tensors in (Cichocki et al., 2016)) is defined as the summation over all the possible values of the repeated indices (*a.k.a.*, modes (Kolda & Bader, 2009)) of a set of core tensors (Orús, 2014). For example, we can use the index contraction to respectively represent the matrix product and the mode-2 product of an order-3 tensor and a matrix:

$$\mathbf{C} = \sum_{r=1}^{R} \mathbf{A}(:, r)\mathbf{B}(:, r) \quad \text{and} \quad \mathcal{Z} = \sum_{r=1}^{R} \mathcal{X}(:, r, :)\mathbf{Y}(:, r), \tag{1}$$

where we use the Matlab® syntax $\mathbf{A}(:, r)$ and $\mathbf{X}(:, r, :)$ to denote a piece of a tensor.

Based on above notations, a *tensor network* is roughly defined as a set of low-order tensors (*i.e.*, core tensors) where some, or all, of its indices/modes are contacted according to some patterns (Orús, 2014). For instance, the well-known tensor train (TT) (Oseledets, 2011) and tensor ring (TR) (Zhao et al., 2016) over the given core tensors $\mathcal{V}_i,\ i \in \{1, 2, 3\}$ can be respectively represented by

$$\begin{aligned} \mathcal{X} &= \sum_{r_1, r_2=1}^{R} \mathcal{V}_1(:, r_1)\mathcal{V}_2(r_1, :, r_2)\mathcal{V}_3(r_2, :) \quad \text{and} \\ \mathcal{Y} &= \sum_{r_0, r_1, r_2=1}^{R} \mathcal{V}_1(r_0, :, r_1)\mathcal{V}_2(r_1, :, r_2)\mathcal{V}_3(r_2, :, r_0), \end{aligned} \tag{2}$$

where for simplicity we assume that the contracted indices/modes take the unified R value, and in practice we can choose different values for the summation.

2.2. Graphical Representation of TN

As shown in aforementioned examples, it is not difficult to introduce a graphical representation to describe the contraction operations among indices. Given an undirected graph $G = (V, E)$ we can construct a tensor network, where the vertices are labeled by different symbols (in order to correspond different core tensors) and each edge is assigned with a non-negative integer weight. Figure 2 shows the correspondence of terminology between the graph and TN, and illustrates the corresponding diagram of the mentioned models in Eq. (2).

Similar to (Ye & Lim, 2019), we also assume that G is always a simple graph. It implies that there exists no self-loop or multiple edges. Furthermore, we impose additional rules into the model to guarantee the bijection relationship:

Graph	*Tensor network*
vertices	cores/core tensors/factors
edges	contracted indices
degree of vertex	number of indices in each core
weight of edge	upper limit of summation
number of edges	number of indices contracted

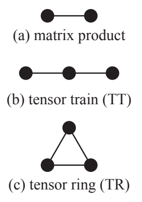


Figure 2. How a graph determines a TN topology (Ye & Lim, 2019). The left table shows the correspondence between graph and TN. Three diagrams on the right illustrate three examples of TNs given in Eq. 2, which includes (a) matrix product, (b) tensor train (TT) and (c)tensor ring (TR), respectively.

Rule 1 *Weight-one edges are not allowed.*

Since dropping the weight-one edges does not change the expression of TN (Ye & Lim, 2019), Rule 1 eliminates the possibility that two different graphs correspond to the same TN structure. Hence, under Rule 1, the correspondence between the TN structure and the graph G becomes one-to-one, which we formalize as

Proposition 1 *Given the number of vertices that are labeled distinctly, there is a bijection from an arbitrary simple graph to the TN structure.*

The proof is straightforward since the non-one weights will change both the order and dimension of the core tensors.

It is implied from Proposition 1 that, with Rule 1, the TN structure can be bijectively embedded into a topological space. Additionally, due to the property of the simple graph, we can uniquely represent the TN structure by its adjacency matrix (Godsil & Royle, 2013), in which the value of each non-diagonal entry equals the weight of the edge or 0 if disconnected. For instance, the TNs *w.r.t.* Eq. (2) can be described by the adjacency matrices,

$$\mathbf{A}_{TT} = \begin{pmatrix} 0 & R & 0 \\ R & 0 & R \\ 0 & R & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{A}_{TR} = \begin{pmatrix} 0 & R & R \\ R & 0 & R \\ R & R & 0 \end{pmatrix}, \tag{3}$$

respectively. Besides Proposition 1, under Rule 1, we can also know the number of all possible structures. Once the number of vertices and weights of edges are fixed, we have

Proposition 2 *Assume that the number of labeled vertices equals N, and the weights of all edges are identical to each other, then there are at least $2^{\frac{N(N-1)}{2}}$ different graphical structures for TN.*

Proposition 2 reveals the difficulty of the topology search problem of TN: The rapidly growing number of possible structures makes it unachievable to seek the optimal solution by enumeration even for a small-scale tensor.

Due to the fact that the connectivity of the graph is not restricted in the model, we should further define how to calculate TN from multiple isolated subgraphs.

Rule 2 *The isolated subgraphs are "connected" by tensor outer product.*

Rule 2 implies that, if there are multiple isolated subgraphs in the TN structure, we first calculate the sub-TN defined by each isolated subgraph using index contraction, then merge all sub-TNs by trivial tensor outer product.

Last, the work in (Ye & Lim, 2019) does not consider the existence of internal cores (*i.e.*, the core tensors whose all indices are connected to other cores), such as the ones in Tucker decomposition and multi-scale entanglement renormalization ansatz (MERA) (Cincio et al., 2008). Hence, below we further allow the existence of the internal cores (vertices), that is,

Rule 3 *The vertices can be internal.*

Remark: The internal vertices are *unlabeled* in contrast to the "external" ones. The existence of multiple internal vertices would theoretically break the aforementioned bijection relationship (Proposition 1), resulting in a surjection from graph to the topological structures of TN. Hence in the rest of the paper, we assume that there exist no more than one internal vertex in the model. While on the practical side, too many internal vertices would lead to the repetitive computation on the same topological structures in our search method. Therefore, we suggest to assume a small number of internal vertices when implementing the method in the paper.

2.3. TN Decomposition and Topology Search

Based on the above definitions about TN and its graphical representation, below we give a general formulation of TN decomposition. Specifically, TN decomposition is defined to represent a tensor $\mathcal{X}$ by a series of index contraction operations on a collection of core tensors $\mathbb{V} := \{\mathcal{V}_i,\, i \in [N]\}$ under a structure *w.r.t.* graph $G = (V, E)$, and it can be generally formulated as (Li et al., 2019)

$$\mathcal{X} = TN(\mathbb{V}; \mathbf{A}), \tag{4}$$

where $TN(\,\cdot\,; \mathbf{A})$ denotes the index contraction operations under the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$. Note that the general model (4) can be degenerated into the conventional decomposition like Tucker, TT, TR decomposition when we choose the specific $\mathbf{A}$'s. Hence it is not difficult to see that there exist many TN decomposition models, which might efficiently represent a tensor yet do not belong to any existing TN decomposition model. Such situation gives rise to the following question: *To represent a tensor, does there exist more efficient TN decomposition models than the existing ones like TT and TR, and how to quickly find them?* In this paper, we call the procedure of solving such problem as the topology search of TN, where the notion "topology" represents the graphical structure of a TN decomposition model. Below, we give a formal statement of topology search for TN decomposition.

Problem 1 *Given a tensor $\mathcal{X}$, the topology search of TN is to find the optimal adjacency matrix $\mathbf{A}_0$, such that there exist a set of core tensors $\widehat{\mathbb{V}}$ that satisfies Eq. (4).*

In different contexts, we can endow the notion "optimal" with different meanings, such as the sparsity of graph or the minimum of the largest degree of vertex. In this paper, we consider to seek a graphical structure of TN, which can *use the fewest parameters to represent a given tensor*. In both machine learning and scientific computing, fewer parameters implies not only lower storage and computational cost but also more promising generalization capacity.

3. GA-based Topology Search Method

3.1. Optimization Model

From Eq. (4) and Problem 1 we see that the topology search is equivalent to seeking the optimal adjacency matrix $\mathbf{A}$ under some criteria. Problem 1 can be therefore formulated as an optimization model:

$$\min_{\mathbf{A}\in\mathbb{A}} \frac{1}{\epsilon\left(\mathbf{A}\right)}, \quad s.t. \quad \left\| \mathcal{X} - TN(\widehat{\mathbb{V}}; \mathbf{A}) \right\|_F^2 \leq \delta, \text{ for some } \widehat{\mathbb{V}}, \tag{5}$$

where $\mathbb{A}$ denotes the set that contains all possible adjacency matrices of a given size, $\|\cdot\|_F$ is the Frobenius norm of a tensor, and $\epsilon(\mathbf{A})$ denotes the compression ratio of the TN decomposition under the given $\mathbf{A}$. It is defined as

$$\epsilon(\mathbf{A}) = \frac{\text{Uncompressed size of } \mathcal{X}}{\text{Parameter size of } \mathbb{V} \text{ under } \mathbf{A}}. \tag{6}$$

As shown in Eq. (5), we maximize the compression ratio, under which the tensor $\mathcal{X}$ can be decomposed into a collection of cores $\mathcal{V}_i$, $\forall i$. Compared to Problem 1, we further impose a tuning parameter δ to provide a tolerance of the noise. It is because in machine learning problems, we often demand a good TN approximation rather than the exact decomposition, and the latter can be accomplished by setting $\delta = 0$.

We can also see that Eq. (5) is a combinatorial optimization model when the weights of the edges are fixed. However, as shown in Proposition 2, the exhaustive search for such problem is unacceptable even for a small-scale problem. To tackle it, we concern a practical alternative, where we are looking for near-optimal solutions that practically well-perform, instead of considering the worst-case behavior of the problem.

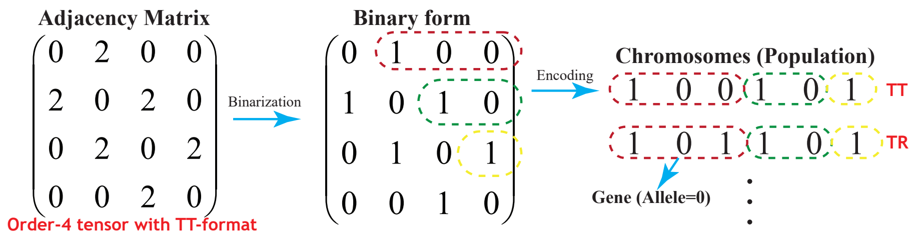


Figure 3. An example to illustrate how to encode an order-4 tensor with TT-format into a binary string *a.k.a* chromosome. First, the adjacency matrix is converted into a binary form; Subsequently, the upper triangle aspect of the matrix is encoded as a binary string. In the context of GA, we generally consider the binary strings as chromosomes or population and their entries as genes.

Algorithm 1 Genetic meta-algorithm for topology search of TN

```
Input: Tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_d}$; Number of of the vertices $N \geq d$ and labeled vertices by the physical modes of $\mathcal{X}$ or internal.
    1. Parents Initialization. //Each individual corresponds to different TN structures (i.e., graphs).
  Iteration until convergence:
    2. Fitness evaluation //Use compression ratio and approximation error of the model to evaluate the graphs.
    3. Elimination //Remove the graphs with bad fitness.
    4. Recombination //Transmit the structures of the superior graphs into the next generation.
    5. Mutation //Slightly change each graph for the diversity of the solutions.
```

There are mainly two potential classes of algorithms in the existing studies to solve Eq. (5), *i.e.*, heuristic solvers and approximation algorithms. For the latter, they generally provide theoretical guarantees, but their scalability may be limited and algorithms with satisfactory bounds may not exist (Arora & Barak, 2009). Therefore, in this work we focus on a heuristic genetic algorithm (GA) to solve the topology search problem for practical concerns.

3.2. Topology Encoding and Meta-algorithm

To do so, we first encode TN's topology into a fixed-length binary string, and then develop a genetic meta-algorithm for the search problem. For simplicity, in the rest of the paper, we assume that the number of vertices is fixed, and the weights on edges are known and identical to each other. Proposition 2 shows that even in this simplified case, the topology search problem is still challenging.

As known in Eq. (4), the TN topology is uniquely determined by its adjacency matrix, and the further assumption of identical weighting on each edge results in an equivalent binary form of the adjacency matrix. We can therefore encode the TN topology by the upper triangle part (the diagonal entries are omitted) of adjacency matrices due to its symmetry property. Figure 3 shows an example to encode an order-4 TT into a binary string. On the other side, Proposition 1 ensures the existence of different TN structures by modifying arbitrary bit from the binary string. For instance, Figure 3 also shows that the modification of the third *bit* turns TT into a TR format.

Our GA for topology search is developed based on the encoded adjacency matrix as chromosomes. Alg. 1 shows the proposed genetic meta-algorithm. In the algorithm, we use the encoded binary strings to be the individuals, and all the genetic operators are directly employed on them. As shown in Alg. 1, the steps 2-3 simulate the procedures of "natural selection", while steps 4-5 guarantee the generation of new individuals. From the view of optimization, steps 2-3 accelerate the convergence to a local optimal value, while steps 4-5 push the approach to explore new solutions to achieve the global optimal values.

3.3. Heuristic Genetic Operators

Below, we present a group of simple genetic operators used in our experiments, and the experimental results demonstrate the operators' effectiveness. Note that, as a meta-algorithm, we expect that various well-developed GA's can be applied to solving this problem. Redundant discussion on detailed tricks in GA is out of the scope in this paper.

Parent initialization. Given the number of vertices N, we initialize a set of randomized individuals (binary strings). For each individual, we first randomly generate a parameter p at uniform distribution on $[0, 1]$, and then each bit of the individual is independently sampled from a Bernoulli distribution $\mathrm{B}(p)$. The randomly generated probability p for the Bernoulli distribution can make the individuals uniformly spread in the solution space.

Fitness evaluation. We evaluate the fitness for each individual by the following function:

$$F(\widehat{\mathbf{A}}) = \underbrace{\frac{1}{\epsilon(\widehat{\mathbf{A}})}}_{\text{compression ratio (CR)}} + \underbrace{\lambda \cdot \min_{\widehat{\mathbb{V}}} \left\| \mathcal{X} - TN(\widehat{\mathbb{V}}; \widehat{\mathbf{A}}) \right\|_F^2 / \|\mathcal{X}\|_F^2}_{\text{relative square error (RSE}^2\text{)}}, \tag{7}$$

where $\widehat{\mathbf{A}}$ denotes the adjacency matrix *w.r.t.* the individual. Compared to Eq. (5), we put the constraint into the function. Since $\|\mathcal{X}\|_F^2$ is generally a non-zero constant in practice, there always exists a λ, such that minimizing Eq. (7) has the same solution to Eq. (5). Furthermore, we can see that Eq. (7) can be split into two interpretable terms, *i.e.* compression ratio (CR) and relative square error (RSE). It hence implies that our method seeks a simple TN model yet with good representation on $\mathcal{X}$, and the tuning parameter λ reflects the trade-off between the two properties.

Elimination. To speed up the convergence, we hope that the individuals in the new generation only inherit the structures from the superior parents in the elder generation. Therefore, in the operator, we eliminate $m\%$ individuals that have the worst fitness in the population.

Recombination and mutation. For recombination, we perform a Russian roulette process to determine which individuals to be the parents. Each parent is determined by a non-uniformly sampling over the set of survival individuals, and the probability for each individual is heuristically given by[2]

$$Pr = \max\left\{0.01, \ln\left(\frac{\alpha}{eps + \beta \cdot rank}\right)\right\}, \tag{8}$$

where $\ln(\cdot)$ denotes the natural logarithm, $rank$ denotes the rank of the individuals by fitness evaluation, eps denotes a small value for numerical stability, and tuning parameters $\alpha, \beta > 0$. In Eq. (8), the constant 0.01 is to avoid the domination of elite individuals. Parameters α, β control the concentration of the probability, which balances the trade-off between the convergence rate and the solutions' variety. After determining the parents, we randomly exchange half of their bits for recombination (*a.k.a.*, crossover).

After recombination, the mutation process of an individual involves flopping each bit independently with a small probability (like 0.05).

4. Experimental Results

In the experimental analysis section, we employ both synthetic and real-world data to demonstrate the effectiveness and efficiency of the method.

4.1. Implementation

In the experiments, we implement our GA on graphics processing unit (GPU, Nvidia® V100) clusters following a central processing unit (CPU, Intel® Xeon® E5-2690) node. Concretely, we exploit the CPU node for receiving the data, employing all genetic operators and assigning the individuals into different GPUs, which calculate the TN decomposition under given topology and output the fitness value. After the calculation for each generation, the CPU node will collect the fitness values and generates new individuals for the next generation.

4.2. Synthetic Data

Below, we employ the synthesized data to examine whether the proposed GA can acquire sufficiently good topological structures.

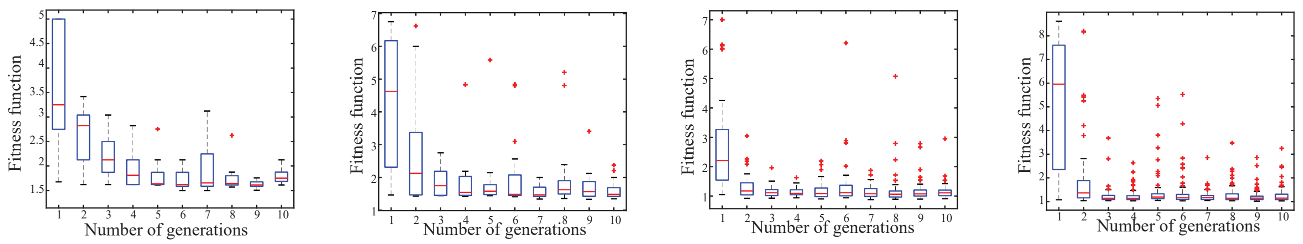
Setup. We first generate batches of tensors with randomized topological structures. The order of the tensors are selected from $\{4, 5, 6, 7, 8\}$, and for each order we generate 5 different samples for evaluation (25 in total). We control the edges of the graph by flipping a coin with a randomized probability $p \in [0.15, 0.85]$ to be the head (connected). Then we initialize the core tensors, of which the entries obey Gaussian distribution with zero mean and 0.1 standard deviation, and calculate the value of their represented tensors. Using the generated tensors, we employ the proposed GA to search the (near-)optimal topology.

We specify the maximum number of the generations for all runs to be 30. The population in each generation are set to be 50 for the ground-truth with order $\{4, 5, 6, 7, 8\}$, respectively. To balance the scale between the compression ratio and RSE, we adjust the trade-off parameter λ in computing the fitness score to be 50. During each generation in GA, 20% of the individuals with the worst fitness scores are eliminated. Meanwhile, to calculate the selection probability described in Eq. (8), we choose the hyper-parameter $\alpha = 200, \beta = 5$. Moreover, we deploy a chance of 5% for each connection to mutate to the opposite state after the recombination is finished. We follow the differentiable programming approach (Liao et al., 2019) for computation of the RSE in Eq. (7). Concretely, for each individual, we initialize the core tensors with Gaussian distribution of zero mean and 0.1 standard deviation, and apply the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 0.001 to carry out the gradient descent steps. In order to avoid the local minima during the TN decomposition, we repeat the decomposition 4 times for each individual under different initialization, and select the smallest RSE for fitness evaluation.

[2]In the formula, we omit the normalization for brevity.

Table 1. Experimental results of representation of synthetic data. The optimal individuals are selected as the first individuals that have an RSE smaller than 10^{-3} with fewest parameters among all the generations. **Gen.** denotes the generation of the optimal individuals. **Eff.** denotes the parameter ratio between the ground-truth TNs and optimal individuals, 1.000 stands for the number of parameters of the two structures are the same, and a larger **Eff.** indicates fewer parameters of TN obtained by our method.

Trial	**Order**									
	4		**5**		**6**		**7**		**8**	
	Gen.	**Eff.**	**Gen.**	**Eff.**	**Gen.**	**Eff.**	**Gen.**	**Eff.**	**Gen.**	**Eff.**
A	001	1.333	003	1.084	005	1.052	003	1.052	001	1.396
B	001	1.500	003	1.690	006	1.062	018	1.000	005	1.000
C	001	1.500	002	1.000	004	1.000	003	1.000	001	1.320
D	001	1.000	001	1.445	003	1.000	001	1.000	020	1.000
E	001	1.000	002	1.000	005	1.052	003	1.000	005	1.000



(a) Order-5 (b) Order-6 (c) Order-7 (d) Order-8

Figure 4. Convergence of our GA. The plots illustrate the distributions of the fitness *w.r.t.* the individuals during the first 10 generations in Trial A. A smaller fitness score suggests the individual has fewer parameters or smaller RSE.

Table 2. Comparison of the number of the required calculated candidates between the trivial enumeration and our GA, where the results in GA is obtained by averaging the number of evaluated individuals in the experiment.

Order	**Enumeration**	**GA**
4	64	18
5	1024	110
6	32768	156
7	2097152	510
8	268435456	1400

Results. The experimental results are reported in Table 1. Under each order configuration, we denote the 5 samples as Trial A∼E, and report when the optimal individuals are found and how good their structures[3] are. Specifically, the **Gen.** columns indicate the generations in which the optimal individuals appear, and the **Eff.** columns indicate the ratio of number of parameters between the ground-truth TNs and discovered optimal individuals, where 1.000 stands for the number of parameters of the two structures are the same, and the larger **Eff.** values indicate the individuals own fewer parameters than its ground-truth TN. It can be seen that, our proposed GA achieves to acquire tensor network structures that are identical or equivalent or even more compact form to the structures of ground-truth structures.

Figure 4 illustrates the convergence of our method. We depict the distributions of the fitness *w.r.t.* the individuals in the first 10 generations under Trial A. Although it appears that there always exist outliers due to the mutation, the median of the fitness drops rapidly in the first several generations. As shown in the **Gen.** columns of Table 1, most of the solutions can be obtained within 5 generations when the tensor order is small (≤ 6), and the required number of generations increases when the order is large. Furthermore, we compare the numbers of the evaluated individual with a trivial enumeration as shown in Table 2. Apparently, the proposed GA efficiently find the good solution from a huge number of candidates.

4.3. Natural Images

Below, we exploit the real-world data (natural images) to evaluate the representational power of TN decomposition

[3]The detailed structures of the TNs are provided in the supplementary material.

Table 3. Experimental results of approximation of real world data. We compare the log compression ratio of GA (ours), TT-SVD, TR-SVD under close RSE value. In GA, the RSE between the original and reconstructed images are obtained from the optimal individuals with their weights setting to 6 and 7. A larger log compression ratio indicates fewer parameters of the model while a smaller RSE indicates better approximation quality.

Images	**Log compression ratio (CR)↑ + RSE↓ – *CR(RSE)***					
	GA(weights=6)	**TT**	**TR**	**GA(weights=7)**	**TT**	**TR**
0	0.901(*0.137*)	0.582(*0.142*)	0.469(*0.141*)	0.660(*0.115*)	0.325(*0.115*)	0.457(*0.127*)
1	1.352(*0.158*)	1.210(*0.170*)	1.216(*0.187*)	1.159(*0.155*)	1.137(*0.166*)	0.824(*0.155*)
2	1.452(*0.176*)	1.148(*0.187*)	1.231(*0.206*)	1.268(*0.171*)	0.898(*0.179*)	1.022(*0.182*)
3	1.649(*0.193*)	1.140(*0.191*)	1.416(*0.211*)	1.476(*0.189*)	1.265(*0.206*)	1.074(*0.191*)
4	0.859(*0.152*)	0.527(*0.156*)	0.403(*0.153*)	0.621(*0.121*)	0.408(*0.143*)	0.372(*0.141*)
5	1.726(*0.087*)	1.471(*0.087*)	1.471(*0.088*)	1.548(*0.083*)	1.531(*0.083*)	1.388(*0.085*)
6	1.332(*0.110*)	1.471(*0.113*)	1.212(*0.124*)	1.141(*0.104*)	1.088(*0.101*)	1.052(*0.102*)
7	1.573(*0.126*)	1.030(*0.139*)	1.112(*0.145*)	1.406(*0.120*)	1.179(*0.142*)	0.970(*0.125*)
8	1.679(*0.085*)	1.493(*0.082*)	1.387(*0.085*)	1.505(*0.081*)	1.493(*0.082*)	1.357(*0.084*)
9	1.164(*0.194*)	0.994(*0.227*)	0.836(*0.200*)	0.966(*0.185*)	0.774(*0.190*)	0.916(*0.226*)

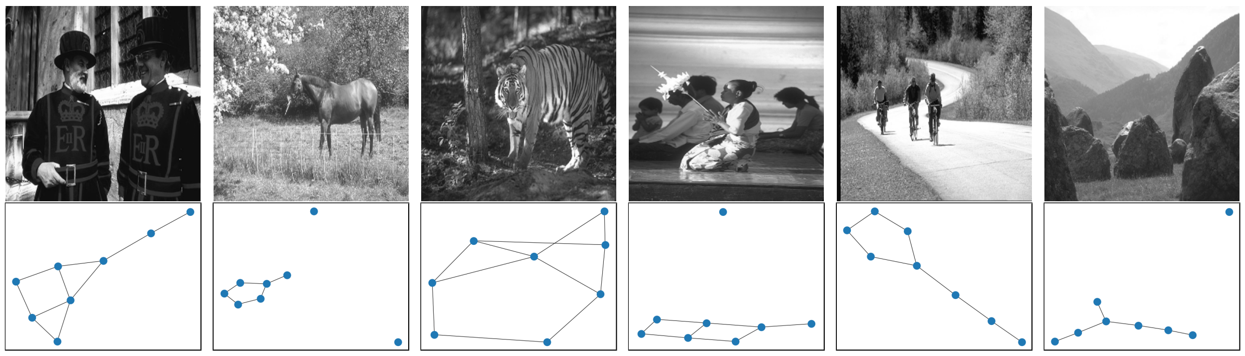


Figure 5. Example to illustrate the employed images and their corresponding TN topological structures obtained by GA.

with the discovered topology by GA.

Setup. In the experiment, we randomly select 10 natural images from the LIVE dataset (Sheikh et al., 2006) and apply TN decomposition to the data approximation task. The images have original sizes of 256×256 and are tensorized to order-8 of the size 4^8. For topology search, we spawn a group of individuals with population 200 in each generation, and set the maximum number of generations to be 15. In addition, the weight of each edge is equal to 4, and we set $\lambda = 1$ and $\beta = 1$. Other parameters in GA are same to the ones given in Sec. 4.2.

Also we implement TT-SVD (Oseledets, 2011) and TR-SVD (Zhao et al., 2016) in the experiment for comparison. In their methods, we manually adjust the tolerance value to meet RSE obtained by our methods. In ours, we change the weights to be $\{6, 7\}$ after topology search to simulate different approximation errors.

Results. The experimental results of compression ratio (CR) (in $\log$ form) with the corresponding RSE are given in Table 3. As shown in Table 3, our methods result in less number of parameters (higher compression ratio) compared to TT-SVD and TR-SVD in over all data and settings. It implies that the discovered topological structures by GA can provide stronger representational power to TN decomposition than the simple line (TT) and cycle (TR) structures. Figure 5 gives some examples of the topological structures obtained by GA. We can see that the obtained topology shows complex structures, which are generally a combination of lines, cycles and isolated points. Such results are expected because the natural images reflect complicated object relationship and abundant information.

Toy experiment shown in Figure 1. The experimental results shown in Figure 1 are obtained with the similar settings as above, where we use the colorful "lena" of the size $256\times256\times3$ as the data and reshape it as an order-9 tensor. The results in Figure 1 also show the significant advantage of "new" topological structures compared to its simple counterparts.

5. Discussion and Concluding Remarks

Computational complexity. Although we have preferred applying the plain GA to the topology search and the convergence is quite fast, the computational requirement is still dramatically larger than TT-SVD and TR-SVD. The bottleneck is mainly from the gradient-based decomposition algorithm and repeated calculation of the decomposition for each individual.

Optimal solution? There is no guarantee in our GA to obtain the optimal solution. Yet the experimental results demonstrate the effectiveness of the proposed work even though the structures we obtain are near-optimal.

Potential applications. TN has been widely used in the deep learning community (Wang et al., 2018; Yu et al., 2017; Kuznetsov et al., 2019; Hou et al., 2019; Huang et al., 2020; Yang et al., 2017; Pan et al., 2019), but the topological structures used in their methods are assumed to be known and fixed. Hence, we expect that searching more promising structures for TN may further improve the performance in those tasks.

Summary. One aim of this paper is to highlight the importance of the topology search for TN decomposition. We argue based on the empirical results that choosing a suitable topological structure could significantly boost the representational power of TN decomposition, and near-optimal solutions are sufficient to show their advantages compared to simple structures.

Acknowledgements

We thank our (ex-)colleagues Dr. Qibin Zhao for assistance to initiating the problem, and Prof. Andrzej Cichocki (Skoltech), Prof. Takayuki Okatani (RIKEN-AIP/Tohoku Univ.), Prof. Isao Ishikawa (Ehime Univ.), Dr. Masahiro Ikeda, and our friend Dr. Xiao-Yang Liu (Columbia Univ.) for comments that greatly improved the manuscript. This work was partially supported by JSPS KAKENHI (Grant No. 20K19875, 20H04249).

References

Anandkumar, A., Ge, R., Hsu, D., Kakade, S. M., and Telgarsky, M. Tensor decompositions for learning latent variable models. *The Journal of Machine Learning Research*, 15(1):2773–2832, 2014.

Arora, S. and Barak, B. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

Bahadori, M. T., Yu, Q. R., and Liu, Y. Fast multivariate spatio-temporal analysis via low rank tensor learning. In *Advances in Neural Information Processing Systems*, pp. 3491–3499, 2014.

Bellman, R. Dynamic programming. *Science*, 153(3731): 34–37, 1966.

Bishop, C. M. *Pattern recognition and machine learning*. Springer, 2006.

Bro, R. Parafac. tutorial and applications. *Chemometrics and Intelligent laboratory Systems*, 38(2):149–171, 1997.

Calabrese, P., Mintchev, M., and Vicari, E. The entanglement entropy of one-dimensional systems in continuous and homogeneous space. *Journal of Statistical Mechanics: Theory and Experiment*, 2011(09):P09028, 2011.

Cheng, S., Wang, L., Xiang, T., and Zhang, P. Tree tensor networks for generative modeling. *arXiv preprint arXiv:1901.02217*, 2019.

Cichocki, A. Era of big data processing: A new approach via tensor networks and tensor decompositions. *arXiv preprint arXiv:1403.2048*, 2014.

Cichocki, A., Lee, N., Oseledets, I., Phan, A.-H., Zhao, Q., Mandic, D. P., et al. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends R in Machine Learning*, 9(4-5):249–429, 2016.

Cincio, L., Dziarmaga, J., and Rams, M. M. Multiscale entanglement renormalization ansatz in two dimensions: quantum ising model. *Physical Review Letters*, 100(24): 240603, 2008.

De Lathauwer, L. Decompositions of a higher-order tensor in block terms—part I: Lemmas for partitioned matrices. *SIAM Journal on Matrix Analysis and Applications*, 30 (3):1022–1032, 2008.

Elsken, T., Metzen, J. H., and Hutter, F. Efficient multi-objective neural architecture search via lamarckian evolution. *arXiv preprint arXiv:1804.09081*, 2018.

Elsken, T., Metzen, J. H., and Hutter, F. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

Fannes, M., Nachtergaele, B., and Werner, R. F. Finitely correlated states on quantum spin chains. *Communications in Mathematical Physics*, 144(3):443–490, 1992.

Godsil, C. and Royle, G. F. *Algebraic graph theory*, volume 207. Springer Science & Business Media, 2013.

Grasedyck, L. Hierarchical singular value decomposition of tensors. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2029–2054, 2010.

Harik, G. R., Lobo, F. G., and Goldberg, D. E. The compact genetic algorithm. *IEEE transactions on evolutionary computation*, 3(4):287–297, 1999.

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.

Hillar, C. J. and Lim, L.-H. Most tensor problems are NP-hard. *Journal of the ACM (JACM)*, 60(6):45, 2013.

Hou, M., Tang, J., Zhang, J., Kong, W., and Zhao, Q. Deep multimodal multilinear fusion with high-order polynomial pooling. In *Advances in Neural Information Processing Systems*, pp. 12113–12122, 2019.

Huang, H., Liu, Y., and Zhu, C. Tensor grid decomposition with application to tensor completion. *arXiv preprint arXiv:1903.04735*, 2019.

Huang, Z., Li, C., Duan, F., and Zhao, Q. H-OWAN: Multi-distorted image restoration with tensor 1x1 convolution. *arXiv preprint arXiv:2001.10853*, 2020.

Hübener, R., Nebendahl, V., and Dür, W. Concatenated tensor network states. *New Journal of Physics*, 12(2): 025004, 2010.

Irani, R. and Nasimi, R. Evolving neural network using real coded genetic algorithm for permeability estimation of the reservoir. *Expert Systems with Applications*, 38(8): 9862–9866, 2011.

Khoromskij, B. N. *Tensor numerical methods in scientific computing*, volume 19. Walter de Gruyter GmbH & Co KG, 2018.

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

Kolda, T. G. and Bader, B. W. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

Kuznetsov, M., Polykovskiy, D., Vetrov, D. P., and Zhebrak, A. A prior of a googol gaussians: a tensor ring induced prior for generative models. In *Advances in Neural Information Processing Systems*, pp. 4104–4114, 2019.

Levine, Y., Yakira, D., Cohen, N., and Shashua, A. Deep learning and quantum entanglement: Fundamental connections with implications to network design. *arXiv preprint arXiv:1704.01552*, 2017.

Li, C., Sun, Z., Yu, J., Hou, M., and Zhao, Q. Low-rank embedding of kernels in convolutional neural networks under random shuffling. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3022–3026. IEEE, 2019.

Liang, J., Meyerson, E., and Miikkulainen, R. Evolutionary architecture search for deep multitask networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 466–473, 2018.

Liao, H.-J., Liu, J.-G., Wang, L., and Xiang, T. Differentiable programming tensor networks. *arXiv preprint arXiv:1903.09650*, 2019.

Liu, Z., Shen, Y., Lakshminarasimhan, V. B., Liang, P. P., Zadeh, A. B., and Morency, L.-P. Efficient low-rank multimodal fusion with modality-specific factors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2247–2256, 2018.

Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y., Deb, K., Goodman, E., and Banzhaf, W. NSGA-net: A multi-objective genetic algorithm for neural architecture search, 2019. URL https://openreview.net/forum?id=B1gIf305Ym.

Maulik, U. and Bandyopadhyay, S. Genetic algorithm-based clustering technique. *Pattern recognition*, 33(9): 1455–1465, 2000.

Mickelin, O. and Karaman, S. Tensor ring decomposition. *arXiv preprint arXiv:1807.02513*, 2018.

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

Orús, R. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014.

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

Pan, Y., Xu, J., Wang, M., Ye, J., Wang, F., Bai, K., and Xu, Z. Compressing recurrent neural networks with tensor ring for action recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4683–4690, 2019.

Phan, A.-H., Cichocki, A., Oseledets, I., Asl, S. A., Calvi, G., and Mandic, D. Tensor networks for latent variable analysis: Higher order canonical polyadic decomposition. *arXiv preprint arXiv:1809.00535*, 2018.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4780–4789, 2019.

Sheikh, H. R., Sabir, M. F., and Bovik, A. C. A statistical evaluation of recent full reference image quality assessment algorithms. *IEEE Transactions on Image Processing*, 15(11):3440–3451, 2006.

Sidiropoulos, N. D., De Lathauwer, L., Fu, X., Huang, K., Papalexakis, E. E., and Faloutsos, C. Tensor decomposition for signal processing and machine learning. *IEEE*

Transactions on Signal Processing, 65(13):3551–3582, 2017.

Stoudenmire, E. and Schwab, D. J. Supervised learning with tensor networks. In *Advances in Neural Information Processing Systems*, pp. 4799–4807, 2016.

Tucker, L. R. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

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.

Wang, W., Sun, Y., Eriksson, B., Wang, W., and Aggarwal, V. Wide compression: Tensor ring nets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9329–9338, 2018.

Whitley, D. A genetic algorithm tutorial. *Statistics and Computing*, 4(2):65–85, 1994.

Yang, Y., Krompass, D., and Tresp, V. Tensor-train recurrent neural networks for video classification. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3891–3900. JMLR. org, 2017.

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

Yokota, T., Erem, B., Guler, S., Warfield, S. K., and Hontani, H. Missing slice recovery for tensors using a low-rank model in embedded space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8251–8259, 2018.

Yu, R., Zheng, S., Anandkumar, A., and Yue, Y. Long-term forecasting using higher order tensor rnns. *arXiv preprint arXiv:1711.00073*, 2017.

Zadeh, A., Chen, M., Poria, S., Cambria, E., and Morency, L.-P. Tensor fusion network for multimodal sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1103–1114, 2017.

Zhang, H., Sharan, V., Charikar, M., and Liang, Y. Recovery guarantees for quadratic tensors with sparse observations. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 3322–3332, 2019.

Zhao, Q., Zhou, G., Xie, S., Zhang, L., and Cichocki, A. Tensor ring decomposition. *arXiv preprint arXiv:1606.05535*, 2016.