Towards a polynomial algorithm for optimal contraction sequence of tensor networks from trees

Jianyu Xu, Ling Liang, Lei Deng, Changyun Wen, Yuan Xie, and Guoqi Li

Phys. Rev. E **100**, 043309 — Published 31 October 2019

DOI: 10.1103/PhysRevE.100.043309

Towards a Polynomial Algorithm for Optimal Contraction Sequence of Tensor Networks from Trees

Jianyu Xu[1], Ling Liang[2], Lei Deng[2†], Changyun Wen[3], Yuan Xie[2], Guoqi Li[1†]
[1]Department of Precision Instrument, Center for Brain Inspired
Computing Research, Tsinghua University, Beijing 100084, China.
[2]Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106, USA.
[3]School of EEE, Nanyang Technological University, Singapore, 639798.
[†] Corresponding to liguoqi@mail.tsinghua.edu.cn or leideng@ucsb.edu

The computational cost of contracting a tensor network depends on the sequence of contractions, but to decide the sequence of contractions with a minimal computational cost on an arbitrary network has been proved to be an NP-Complete problem. In this work, we conjecture that the problem may be a polynomial one if we consider the computational complexity instead. We propose a polynomial algorithm for the optimal contraction complexity of tensor tree network, which is a specific and widely-applied network structure. We prove that for any tensor tree network, the proposed algorithm can achieve a sequence of contractions that guarantees the minimal time complexity and a linear space complexity simultaneously. To illustrate the validity of our idea, numerical simulations are presented that evidence the significant benefits when the network scale becomes large. This work will have great potential for the efficient processing of various physical simulations and pave the way for the further exploration of the computational complexity of tensor contraction on arbitrary tensor networks.

Key Words Tensor Contraction, Tensor Tree Network, Polynomial Algorithm, Computational Complexity

I. INTRODUCTION

Tensor network [1] plays an important role in the fields of quantum mechanics [2], multi-dimensional data [3], and artificial intelligence [4], with applications involved in strong quantum simulations [5], quantum many-body systems [6], matrix product states (MPS) and projected entangled pair states (PEPS) [7–13], multiscale entanglement renormalization ansatz (MERA) [14, 15], quantum chemistry [16], quantum circuit simulation [17, 18], neural network and machine learning algorithm [4, 19–24] and signal processing [25].

However, there does not exist a unified definition for tensor networks among the articles mentioned above. Although the main idea of mapping a cluster of tensors onto a graph module is valid in these definitions, the ways in which vertices and edges are drawn are different from one another among these works. For instance, in [1] and [14] vertices and edges are used to represent tensors and indices, respectively, but in [17] the condition turns over. According to the convenience of evaluating the time and space complexity, we adopt the definitions in [1] and [14], and make some modifications including assigning weights to edges for representing the number of common indices and assigning weights to vertices for representing the number of free indices. Formal definitions would be presented in the following sections.

Tensor contraction is the basic and most important calculus [26] for tensors. To contract two tensors means to carry out inner product (multiplication in sequence and summation by indices) of some corresponding orders, and perform outer product of the other orders. For instance, a matrix product is a 1-order contraction of two 2-order tensors. This operation is especially important for a tensor network [3, 17, 18, 27–30], because to calculate, or actually contract, a tensor network asks for contractions of pairs of tensors until the whole network is contracted into a single tensor.

Considering the high order and high dimension (the number of elements in each order) of tensors, it is always time-and-space consuming to contract two tensors. As a result, we hope to optimize the efficiency of contraction. Many works contribute to the efficiency of tensor contraction, including a so-called tensor contraction engine, or TCE [16, 31–33] and other versions [34]. The work in [35] has pointed out that the sequence of contractions in a tensor network determines the computational cost(the number of multiplications of elements) of the whole contraction procedure, and it has also proved that the problem to find out the optimal sequence towards the fewest multiplications is NP-Complete. Therefore, most of existing works target at optimizing the efficiency of searching [27, 29] instead of designing a deterministic algorithm. These searching algorithms can actually find out a sequence in which the computational cost is much lower than their baselines, but they cannot ensure an optimal sequence in polynomial time.

Now that the sequence for optimal computational cost is proved to be an NP-Complete problem according to [35], we have to weaken the conditions in order to obtain a deterministic and polynomial algorithm. On one hand, there exist other indicators, the **time complexity** and **space complexity**, that can reflect the cost from the perspective of time and storage. The computational cost in [35] refers to the total number of multiplications of scalars in the whole process of contracting a tensor network, while the computational complexity represents the largest complexity among every contractions of pairs of tensors, or so-called "steps". If we suppose that the total number of tensors does not increase with the dimension, i.e. the number of elements in each order, of every tensor, we prove that the computational cost in [35] is equivalent to the time complexity of the same process under the function of $O(\cdot)$.

On the other hand, it is rational for us to conjecture that it is easier to determine the sequence for a minimal time(and space) complexity than to determine that for a minimal computational cost, since the calculation of complexity is much simpler than that of cost. This is also because that we prove a subproblem, to determine the sequence for a minimal time and space complexity on a **Tensor Tree Network** (TTN) [36, 37], also termed as Tree Tensor Network State (TTNS) [38], to be *polynomial*, while the problem of achieving the sequence for a minimal computational cost on tensor tree network has neither been determined to be polynomial nor NP-Complete yet. It is worth mentioning that there are many different tensor network structures (see [1–3, 39–41] for a review), and tensor tree network is a typical structure which is both likely to find a polynomial algorithm and significant for many applications. These applications include simulations on quantum systems [42, 43], quantum critical Hamiltons [44], quantum chemistry [37], Bathe lattice [45], spain chains & lattices [46], and interacting fermions [47], referring to [48] in a nutshell. Besides, in tensor decomposition algorithms, the results of Tucker decomposition [49], Hierarchical Tucker decomposition [50], and MPS [7] are also in the format of tensor tree networks. There are also structures similar to a tensor tree network [51]. More importantly, the work [52] reveals that many other structures of tensor networks can be directly transformed into tensor tree networks without any approximation. Therefore, if we can figure out a polynomial-time algorithm that can determine the optimal sequence of contracting any tensor tree network, then we can reduce the computational cost of many problems listed above.

In this work, we transform the module of tensor network to another module of graph theory, and accordingly transform the problem of minimizing the computational complexity to a problem of optimizing a specific status occurring in a series of operations on that graph module. We notice that a tree graph has many excellent properties, including a simple and recursive structure that can be maintained during contractions. Different from those searching and heuristic ones mentioned above, we propose a polynomial-time algorithm for tree-structure network for the first time. The algorithm can achieve a sequence of contractions that is proved to be optimal in time complexity, and would guarantee a linear space complexity according to input or output. For a deterministic Turing Machine, its space complexity is defined on its work tape(and thus the optimal could be logarithm). However, if we also take the input and output tapes into consideration, we know that a linear space complexity in comparison with input and output is also optimal. Therefore, our "optimal space complexity" refers to a "linear space complexity" in the following part. This paper also proves that at least one specific structure of tensor networks has a polynomial-time algorithm that can reach an optimal sequence of contractions. Since tensor tree network is useful in a number of areas, such as quantum simulation and continuum mechanism as aforementioned, our algorithm is essential for high-performance processing in those domains.

To draw a conclusion, we firstly transform the NP-Complete problem of contraction sequence with optimal computational cost to an open problem of contraction sequence with optimal time and space complexities. Under this problem, we then prove an important subproblem of contraction sequence on tensor tree network to be polynomial. The proof is carried out by a polynomial algorithm, which can simultaneously achieve minimal time and space complexities. Finally, we conduct numerical simulations to illustrate the validity and efficiency of our algorithm. The evaluations verify that our transformation of the problem is rational, and the polynomial time complexity is efficient.

In the following parts, we firstly provide some preliminaries and definitions in Section II. After that, we formulate the problem we are aiming at in Section III. In Section IV we propose our algorithm, and prove its optimality. We also propose and prove some basic rules for optimal solutions, which will turn out to be fingerposts for future researches on the optimal sequence of contractions of a complete graph (network). In Section V we present firstly an example of our algorithm on a tensor tree network, and then numerical simulation results where we will compare not only the time and space gap between the found optimal contraction sequence and the vanilla baseline of random sequence, but also the time spending on finding optimal contraction sequence between our algorithm and an existing algorithm, using randomly generated tensors and tensor tree networks. In Section VI the paper is concluded with some discussions.

II. PRELIMINARIES

According to [1], we can similarly define a *tensor* to be a multidimensional array, and the *order* of a tensor to be the number of indices necessary for referring any element in the tensor. Based on these two definitions, we can then define the tensor contraction and the tensor network.

Tensor Contraction. We name the operation *tensor contraction* when we sum over some common indices, each pair of which occurs twice and only twice, of several tensors as inner products while remaining the other indices as outer products, each of which occurs once and only once. We name these common indices *dummy indices*, and the other indices *free indices*.

For example, with $A \in R^{N_a \times N_b \times N_c \times N_d}$, and $B \in R^{N_b \times N_c \times N_e}$, we can define the contraction of A and B as follows:

$$(AB)_{a,d,e} = \sum_{b=1}^{N_b} \sum_{c=1}^{N_c} A_{a,b,c,d} \cdot B_{b,c,e}. \tag{1}$$

Here we use the expression AB to represent the result of contraction of A and B, and similar expressions will be used in the following parts.

We make an assumption that all the tensors involved are "cubic", which means all indices range from 1 to N. In a later part, we will prove that the module with this assumption is also fit for the cases when tensors are not cubic. With this assumption, we can see FIG. 1 to have a direct illustration of the contractions of two tensors.

Tensor Network. In a contraction of many tensors, we use a graph to represent their relationship: For every tensor, we use a vertex A to represent it, and give it a *weight* W_A that equals the number of free indices it has. For every pair of tensors (vertices) A and B, we use an edge E_{A-B} to connect them, and we give it a weight W_{A-B} that equals the number of dummy indices they have in common. For every edge whose weight is 0, we can delete it. We call this network a *tensor network*, usually denoted as T_{net} (or T for a tree structure in the following sections).

Contractions on a Tensor Network. Now we can transform the definition of tensor contraction to a tensor network: (1) We draw another vertex to represent the result of contraction of the selected tensors (vertices) and give this vertex a weight equaling the sum of these vertices' weights. (2) For every edge who has one and only one end in the selected vertices, we move this end to the newly drawn vertex, with its weight unchanged. (3) For any pair of vertices between whom there is more than one

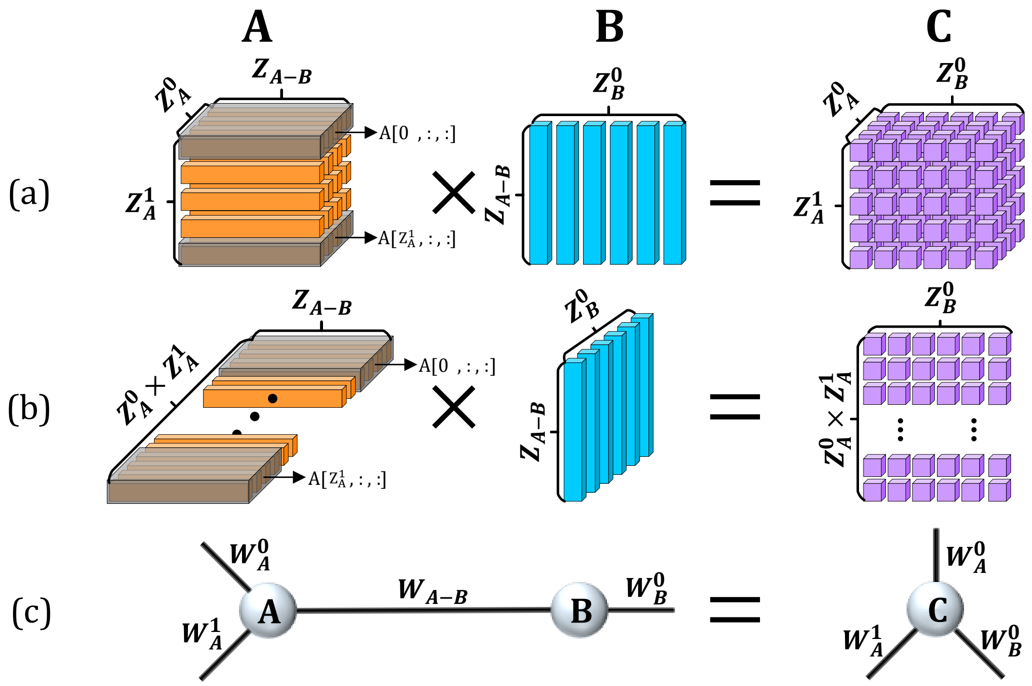

FIG. 1. Illustration of tensor contraction: (a) A contraction of a 3-order tensor A and a 2-order tensor B, with one order R contracted, and the result $C = AB$ is a 3-order tensor; (b) matrix representation of (a), where the 2 non-contracted orders are merged (or "vectorized") into one order and the contraction turns into a matrix product; (c) The equivalent process on a tensor network. Here we adopt $Z_X^i = N^{W_X^i}, i = 0, 1, X = A, B$ and $Z_{A-B} = N^{W_{A-B}}$, where $N \in \mathcal{Z}^+$.

edge, we "merge" these edges into one edge and give this edge a weight equaling the sum of these edges' weights. (4) Erase those selected vertices and edges between them.

Apparently, the definition of *contractions on a tensor network* is equivalent to the definition of *tensor contraction*. Moreover, it is worth pointing out that there are two meanings on contractions associated with a tensor network. Firstly, every pair of connected vertices shares one or more common indices, which has been defined as a contraction. The sharing of indices serves as connections, or edges, between pairs of vertices. Any of these edges would be eliminated in one step of contraction, but before this step it contributes to the structure of network. To this end, contractions serve as the structure of a tensor network. Secondly, in order to calculate(contract) a tensor network, we should contract small groups of vertices (tensors) until there only exists one tensor (vertex). Therefore, contractions also serve as the method to calculate a tensor network.

As defined, the contraction of the whole network can be settled via several contractions of small groups of vertices. Since any contraction of three or more tensors can be broken down to contractions of pairs of tensors sequentially, we stipulate that the tensor contraction refers to that of two tensors without special statements. Accordingly, we also stipulate that the vertex contraction refers to that of two vertices without special statements. In the following sections, we will show that the 2-tensor contraction is capable of achieving optimal time and space complexity. In this way, the tensor network should be contracted by several steps sequentially, with the total number of vertices reduced by one for each step. Therefore, we will define the *sequence of contractions* of a tensor network.

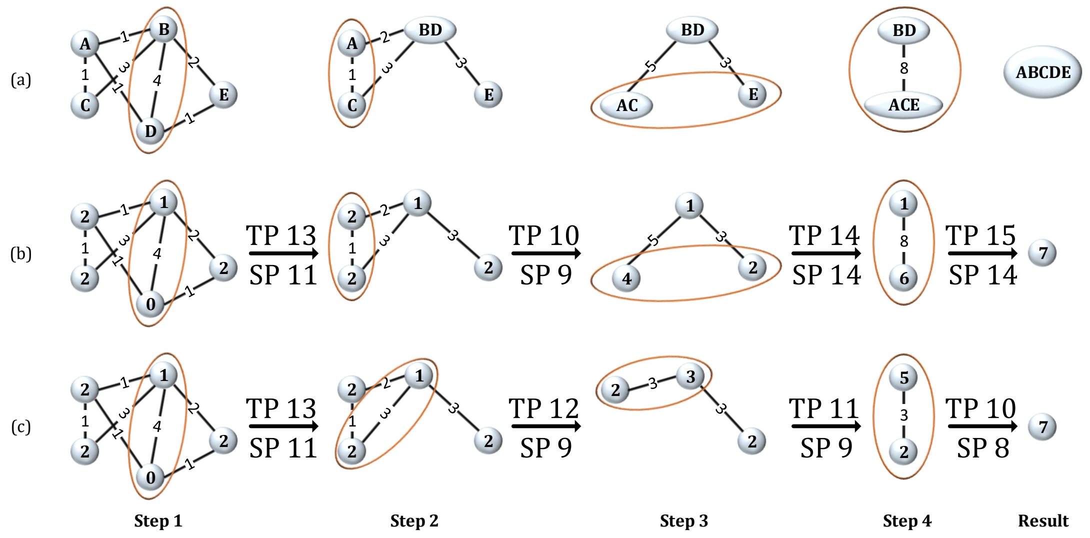

FIG. 2. Contraction of two vertices (tensors) in a tensor network. In sequence (a) we use BD to represent the tensor obtained from the contraction of B and D. The edge between the two tensors is eliminated, and the edges whose one end is either of the two tensors merge these ends onto the new born-in-contraction vertex, with their weights remaining unchanged. Sequence (b) illustrates that the weight of the new vertex equals the sum of weights of the original two vertices. Sequence (b) and (c) are 2 different sequences of contractions, and they have different time and space power: sequence (b) has a time power 15 and space power 14, and those of (c) are 13 and 11 respectively.

Sequence of Contractions. For a tensor network T_{net} with n vertices $V_1, V_2, \dots, V_n$, it turns into T_{net1} with $(n-1)$ vertices after contracting 2 vertices and we denote this step of contraction as the 1^{st} step of contractions, or θ_1. In this way, we similarly get θ_k in contracting tensor network T_{netk-1} into T_{netk}, where $k = 2, 3, \dots, n-1$. By arranging these contractions in order, we can get a sequence of contractions, termed as "sequence of tensor network contractions" which is denoted as $Q_{T_{net}}$. Actually, for any specific sequence we have

$$Q_{T_{net}} = \left\{\theta_1, Q_{T_{net1}}\right\} \tag{2}$$

and

$$Q_{T_{netk-1}} = \left\{\theta_k, Q_{T_{netk}}\right\}, \forall k = 2, 3, \dots, n-1. \tag{3}$$

It is worth mentioning that the sign $Q_{T_{net}}$ is not a function, but a notation for one sequence of contractions of T_{net}.

FIG. 2 illustrates the contraction of a tensor network. In this instance, we aim at calculating the expression

$$\sum_{j,k,l,p,q,r,s,t,u,v,w,x,y} A_{i_1,i_2,j,k,l} \cdot B_{i_3,j,p,q,r,s,t,u,v,w,x} \cdot C_{i_4,i_5,k,p,q,r} \cdot D_{l,s,t,u,v,y} \cdot E_{i_6,i_7,w,x,y} \tag{4}$$

within several steps, which has been shown in FIG. 2 (b). Here every index ranges from 1 to N independently. For instance, we calculate BD in the first step by

$$BD_{i_3,j,l,p,q,r,w,x,y} = \sum_{s=1}^{N}\sum_{t=1}^{N}\sum_{u=1}^{N}\sum_{v=1}^{N} B_{i_3,j,p,q,r,s,t,u,v,w,x} \cdot D_{l,s,t,u,v,y}. \tag{5}$$

For the tensor network at this step, the vertex B is contracted with D. According to the definition of *contraction on network*, we merge E_{A-B} with E_{A-D}, E_{B-C} with E_{C-D} and E_{B-E} with E_{D-E} respectively. For example, we draw E_{A-BD} to substitute E_{A-B} and E_{A-D}, with its weight $W_{A-BD} = W_{A-B} + W_{A-D}$.

In this way we calculate this expression by contracting every pair of tensors. We notice that for each step, the time complexity equals the product of ranges of every index, and the space complexity equals the one with a maximal space complexity of the three tensors. If we suppose that the ranges of every index(order) are all from 1 to N, then the time complexity equals N to the exponential of the number of indices involved. In the following part, we will state these computational complexities, and map them in a graph module as well.

Time Complexity and **Space Complexity**. For tensors $A \in R^{M_1\times M_2\times\ldots\times M_m\times N_1\times N_2\times\ldots\times N_n}$ and $B \in R^{N_1\times N_2\times\ldots\times N_n\times U_1\times U_2\times\ldots\times U_u}$, where $M_r, N_s, U_t \in Z^+, \forall r = 1,\ldots,m; s = 1,\ldots,n; t = 1,\ldots,u$, the time complexity of contracting A with B as the expression

$$\begin{aligned}&(AB)_{i_1,i_2,\ldots,i_m,k_1,k_2,\ldots,k_u}\\ &=\sum_{j_1=1}^{N_1}\sum_{j_2=1}^{N_2}\cdots\sum_{j_n=1}^{N_n}A_{i_1,i_2,\ldots,i_m,j_1,j_2,\ldots,j_n}\cdot B_{j_1,j_2,\ldots,j_n,k_1,k_2,\ldots,k_u}\end{aligned} \tag{6}$$

equals $M_1M_2\ldots M_mN_1N_2\ldots N_nU_1U_2\ldots U_u$, and the space complexity of that equals

$$\begin{aligned}\max\{&M_1M_2\ldots M_mN_1N_2\ldots N_n,\\ &N_1N_2\ldots N_nU_1U_2\ldots U_u,\\ &M_1M_2\ldots M_mU_1U_2\ldots U_u\}.\end{aligned} \tag{7}$$

Specifically, if $M_r = N_s = U_t = N$, the time complexity equals N^{m+n+q} and the space complexity equals $N^{\max\{m+n,n+q,m+q\}}$.

Note: For a sequence of contracting a tensor network, the time/space complexity equals the maximum time/space complexity among every single step.

Simultaneously, we define two functions on the graph: "**Degree**" (denoted as $\mathcal{D}$) of a vertex is defined as the sum of weights connecting to this vertex:

$$\mathcal{D}(A) = \sum_{B\neq A} W_{A-B}; \tag{8}$$

"**Weight and Degree**" (denoted as S_{WD}) of a vertex is the sum of weight and degree of this vertex, i.e.

$$S_{WD}(A) = W_A + \mathcal{D}(A) = W_A + \sum_{B\neq A} W_{A-B}. \tag{9}$$

The S_{WD} value of a vertex equals the exponential of space complexity of the tensor it represents. For example, if there is

$$A \in R^{N_a\times N_b\times N_c\times N_d}, \tag{10}$$

then we can have

$$S_{WD}(A) = log_N N_a + log_N N_b + log_N N_c + log_N N_d. \tag{11}$$

Besides S_{WD}, it is found that any weight of a vertex or an edge on the tensor network equals the exponential, or power, of the element it represents. If we only concern the complexity but not the structure, which means we only consider the number of multiplications of tensors being contracted, we can only count the number of order, or the exponential power, with a fixed base N. This is equivalent to taking logarithms with base N. For an order whose index ranges from 1 to M, we can suppose this order to be $log_N M$ since $M = N^{log_N M}$. Since M and N are all positive integers, the numbers representing weights of vertices and edges on the tensor network can be any non-negative real number. To this end, we can assume that the range of every index(or order) equals N consistently, and that the weight of any edge or vertex can be any positive real number. Next, we define the time and space power of one contraction in a tensor network according to time and space complexity we have stated.

Definition 1 (Time Power) *In a contraction of two vertices, we use P_T to express time power:*

$$P_T(AB) = S_{WD}(A) + S_{WD}(B) - W_{A-B}, \tag{12}$$

where W_{A-B} means the weight of edge connecting A and B.

Definition 2 (Space Power) *In a contraction of vertices, we use P_S to express the space power.*

$$P_S(AB) = \max\{S_{WD}(A), S_{WD}(B), S_{WD}(AB)\}. \tag{13}$$

Note: In this paper, there are several terminologies: computational cost, time and space complexity, time and space power. These terminologies may result in confusion. Therefore, we now clarify these words:

1. The term **computational cost**, or **cost**, refers to the total number of multiplications during the whole process of contractions of a tensor network;
2. The term **time complexity** refers to the maximum time complexity among every single step of contracting a tensor network;
3. The term **space complexity** refers to the maximum space complexity among every single step of contracting a tensor network;
4. The term **time power**, or P_T, refers to the logarithm of time complexity with base N. Equivalently, time complexity equals N^{P_T};
5. The term **space power**, or P_S, refers to the logarithm of space complexity with base N. Equivalently, space complexity equals N^{P_S}.

Besides, in Section IV D we will prove that our algorithm can be conducted in polynomial time complexity. Here the time complexity of the algorithm is not relevant to the complexities of contracting tensor networks.

III. PROBLEM FORMULATION

The contraction of a whole tensor network can be seen as a collection of many single steps of 2-tensor contraction. As mentioned in [18], the original problem is to minimize the total number of multiplications (computational cost). Since this is an NP-Complete problem, we cannot expect to achieve a sequence of contractions via a polynomial algorithm. Therefore, we try to consider a similar problem, i.e. computational complexity, which focuses on the reduction of difficulty. Also, we solve a specified and significant subproblem in this paper: the optimal sequences of contractions of *tensor tree networks*.

Before we define a *tensor tree network*, we firstly define some basic terminologies that we would use in the next sections:

(1) A **path** on a graph is an ordered set of vertices and edges $p(v_0, v_n) = v_0 e_1 v_1 e_2 \dots v_{n-1} e_n v_n$, where $e_i = (v_{i-1}, v_i), \forall i = 1, 2, \dots, n$. We record n as the length of the path $p(v_0, v_n)$.

(2) A **connective** (or **connected**) graph is a graph $G < V, E >$ where $\forall v_i, v_j \in V$ there exists a path $p(v_i, v_j) \in G$.

(3) A **tree** graph is a connective graph with no loop. Also, when the graph G is a tree, there exists one and only one $p(v_i, v_j) \in G, \forall v_i, v_j \in V$.

(4) A **leaf** on a tree is a vertex that connects only one other vertex.

(5) A **root** of a tree is the "origin" vertex of the tree. Actually, any vertex on a tree can be selected as the **root**.

(6) For a tree $G < V, E >$ with a determined root v_0, we can define the relationship of **father vertex** and **child vertex**: for any pair of vertices $v_i, v_j \in V$ that is connected by one edge $v_i v_j$, if the path $p(v_0, vi)$ is shorter than $p(v_0, v_j)$, then v_i is the **father vertex** of v_j, and v_j is the **child vertex** of v_i. Generally, we call the only vertex connecting a leaf the father vertex of the leaf.

(7) For a tree $G < V, E >$ with a determined root v_0, and for any vertex $v_i \in V$, if the length of the path $p(v_0, v_i)$ equals a nonnegative number r, then we say that v_i on the r^{th} **layer**. Specifically, v_0 is on the 0^{th} layer.

(8) For a tree $G < V, E >$ with a determined root v_0, we define the **width** of a tree as the maximal of numbers of vertices among each layers.

(9) For a tree $G < V, E >$, if a tree $G_1 < V_1, E_1 >$ satisfies that $G_1 \subseteq G$, then we call G_1 as a **subtree** of G.

The definition of a **Tensor Tree Network** is that the structure of the tensor network is a tree. Fig. 3 shows a comparison of a typical tensor network with a tensor tree network.

In order to transform the problem, we make the following assumptions:

1. The tensor networks we deal with are *tensor tree networks*;
2. The tensor tree network is supposed to be connected without losing generality;
3. There are n tensors (vertices) in the tensor tree network;
4. We only contract two vertices that are connected at each step;

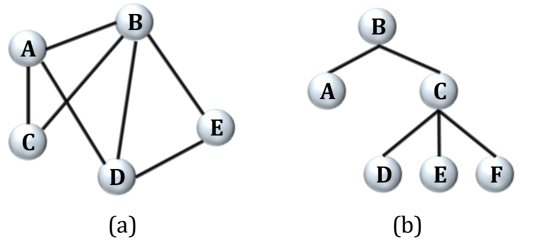

FIG. 3. Comparison of (a) a common tensor network with (b) a tensor tree network. In (a) there exist cycles of connected vertices, such as {A,B,C}, {A,B,D}, {B,E,D}. Different from that, in tensor tree network (b) there does not exist any cycle.

5. The weights of vertices are all non-negative real numbers, and the weights of edges are all positive without losing generality;

6. We consider n to be a **relative** constant to N. This means n does not increase with N while considering the complexity of tensor contractions, but it determines the complexity of algorithm while considering the optimal sequence of contractions.

Besides, we have made "cubic" and "two-tensor-contraction" assumptions in the past sections. As stated in Section I and Section II, the time/space complexity of contracting the whole tensor network equals the **largest** time/space complexity among every step in the sequence of contractions. According to this, the time/space power equals the largest time/space power. These can be denoted as the following equations:

$$P_T\left(Q_{T_{net}}\right) = \max_{k=1,2,\ldots,n-1}\left\{P_T\left(\theta_k\right)\right\}; \tag{14}$$

$$P_S\left(Q_{T_{net}}\right) = \max_{k=1,2,\ldots,n-1}\left\{P_S\left(\theta_k\right)\right\}. \tag{15}$$

Remark 1. We have $O(\textit{computational cost}) = O(\textit{time complexity}) = N^{P_T}$ according to our assumptions. This also means that "cost" and "complexity" are equivalent under our assumptions. This is because that on one hand, computational cost equals the total number of multiplications of every pair of scalars during the whole process of contracting the tensor network, while time complexity equals the number of multiplications of one single step. Therefore, we have $\textit{computational cost} \geq \textit{time complexity}$. On the other hand, since time complexity equals the largest one among every of all steps, we have $\textit{computational cost} \leq n \times (\textit{time complexity})$. Since n is a relative constant to N, we know that $O(\textit{computational cost}) = O(\textit{time complexity})$.

For the assumption of a tree graph, we adopt it because there are no edges merging if we only contract those pairs of vertices that are connected with a non-zero weighted edge. Therefore, the structure of a tree network would maintain a tree under these contractions .

Remark 2. In a connected tensor tree network, and under the stipulation of contracting two connected tensors at each time, we can equivalently consider each step of contracting two tensors as *eliminating* one edge and merging its two ends. We will use the word "eliminate" for several times in the following sections.

Comparing FIG. 2 (b) with (c), we notice that different sequences of contractions may have different time and space power. Therefore, it is an important issue to determine the optimal sequence on time and space power. The problem we will solve in this work is to design an appropriate algorithm that can achieve a sequence of contractions on a tensor tree network with an optimal computational complexity (power) in both time and space. Moreover, the algorithm should be executed in polynomial time complexity, with respect to the number n of tensors (vertices), for any tensor tree network.

Remark 3. For any group of tensors, the time power of contracting them by one pair of 2 vertices for each time would not exceed that of contracting them together at one time. Actually, for the latter one, the time power equals the sum of all weights of edges and vertices involved. However, for every step of the former one, the time power equals the sum of some weights involved, which is a subset of that in the latter one. Therefore, the former time power would not exceed the latter one. This also means that our presuppose of 2-tensor contraction (and 2-vertices contraction) is able to achieve an optimal sequence.

Algorithm 1: Optimal Sequence of Tensor Tree Network Contraction

```
Input: Tensor tree network $T_n$, number of vertices $N$
Output: Sequence of contractions
1  Set $S_1 = \{weights\ \ of\ \ every\ \ edge\}$;
2  Set $S_2 = \{weights\ \ of\ \ every\ \ leaf\}$;
3  Set $S = S_1 \cup S_2$;
4  The Case $n = 1$ and $n = 2$ are trivial;
5  while $n \geq 3$ do
6      if $\min S \in S_2$ then
7          Contract the leaf whose weight is this minimum to its father vertex IN THE
             FIRST STEP;
8          $n = n - 1$ go to step 4;
9      else
10         Determine the edge whose weight is the minimum to be eliminated IN THE
             LAST STEP;
11         Add the weight of the two ending vertices of this edge by this minimum;
12         Divide the tree into 2 subtrees into $T_k$ and $T_{n-k}$;
13         $T_k$ with n=k go to step 4;
14         $T_{n-k}$ with n=n-k go to step 4;
15     end
16 end
```

FIG. 4. Algorithm to find the optimal contraction sequence on tensor tree network. This is an recursive algorithm, which reduces the problem from a scale of n to one or two subproblems with their scale $k < n$ for each iteration. In subsections IV A and IV B, we will sequentially prove the sequence to be optimal in both time and space complexity. In subsection IV C we will prove our stipulation on edge contractions to be rational. Finally, in subsection IV D we prove our algorithm to be polynomial.

IV. ALGORITHM AND ANALYSIS

In this section, we first propose an algorithm to solve the problem described in Section III to obtain the optimal sequence of tensor contractions on a tree network. The algorithm is illustrated in FIG. 4 (Algorithm 1), and FIG. 5 illustrates how our algorithm works in a specific tree structure. Without losing generality, we then describe and analyze the algorithm in the equivalent graph module.

Later we will prove the proposed algorithm actually achieves minima on both time and space power in subsections IV A and IV B sequentially. In Subsection IV C, we will prove that our presuppose for our algorithm to adopt only edge contractions is able to achieve optimal time power by proving that the result would not be better without this presuppose. Finally, in Subsection IV D we will prove that our algorithm can be executed in polynomial time. By this way, we shall strictly prove that the optimally contracting tensor tree network is a Polynomial problem, which provides a solid theoretical foundation for our conjecture in this work.

A. Optimal Time Complexity

In this subsection, we prove that our algorithm can achieve the sequence with optimal time complexity, which is equivalent to achieving an optimal time power on the graph. Before doing this, we firstly present the following lemmas.

Lemma 1 *The minimal weight of leaves would not decrease during contractions of a tree. In other words, for any tensor tree network T, and any one*

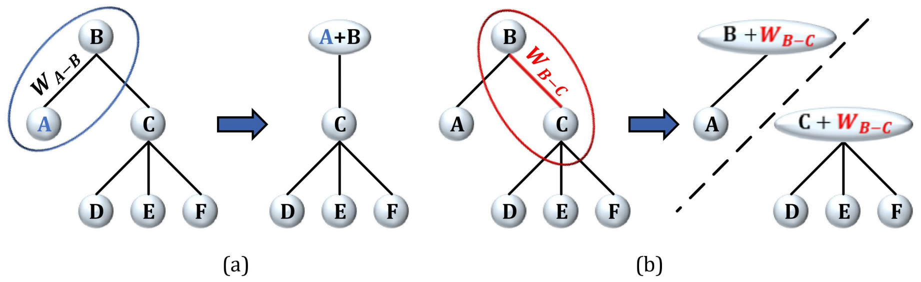

FIG. 5. How our algorithm finds the optimal contraction sequence: (a) The case when A is the smallest element in the set S of weights of leaves and edges. Since it occurs on a leaf, we contract this leaf with its father B in the first step. (b) The case when W_{B-C} is the smallest element in the set S. We determine to contract B with C in the last step. To contract it in the last step means to consider the two parts divided by E_{B-C} separately before the last step.

of the intermediate or final results during contractions, denoted as T', we have

$$\min_{V_k \text{ is a leaf of } T} W_{V_k} \le \min_{V_k' \text{ is a leaf of } T'} W_{V_K'}. \tag{16}$$

Proof. Before the final step, contractions of a tree can be divided into the following two cases: (i) a leaf contracts with a non-leaf vertex, and (ii) a non-leaf vertex contracts with another.

The latter case cannot emerge a leaf. This is because both of them have at least one edge other than their common one, and according to the property of a tree, these two edges are not connected with each other. Therefore, the vertex born in their contraction has at least two edges, which determines it a non-leaf vertex.

As a result, a leaf born in contraction must come out from another leaf. Since weights of vertices are summed up during contraction, the weight of newly-born leaf cannot be less than that of an existing leaf. Therefore, the minimum of weights of leaves cannot decrease.■

Lemma 2 *For a tensor tree network T, we suppose the edge E_{A-B} be eliminated in θ_{n-1}, and the tree be divided into 2 independent subtrees T_A and T_B by this edge, whose roots are A and B respectively. Therefore, we have:*

$$\begin{aligned}&P_T(Q_T)\\ &\ge \max\left\{\min_{Q_{T_A}}\left\{P_T\left(Q_{T_A}\right)\right\}, \min_{Q_{T_B}}\left\{P_T\left(Q_{T_B}\right)\right\}, P_T(\theta_{n-1})\right\}\\ &=P_T\left(\left\{\arg\min_{Q_{T_A}}\left\{P_T\left(Q_{T_A}\right)\right\}, \arg\min_{Q_{T_B}}\left\{P_T\left(Q_{T_B}\right)\right\}, \theta_{n-1}\right\}\right).\end{aligned} \tag{17}$$

Proof. We prove it by contradiction. We name our sequence of contractions, to optimally contract T_A and T_B and finally eliminate E_{A-B}, as "Q_1". Suppose there exists another sequence, named as "Q_2", whose last step is to eliminate E_{A-B} and whose time power is less than that of Q_1.

Assume the time power of contracting T_A in Q_1 to be $P_T(Q_{1A})$, and that in Q_2 to be $P_T(Q_{2A})$. Similarly we can get $P_T(Q_{1B})$ and $P_T(Q_{2B})$. Since the final step in the two sequences are the same, we suppose the time power of this step to be $P_{T\,final}$.

According to the definition of time power of a sequence of contractions, we know that the time power of Q_1 and Q_2 are

$$\begin{cases}\max\left\{P_T(Q_{1A}), P_T(Q_{1B}), P_{T\,final}\right\},\\ \max\left\{P_T(Q_{2A}), P_T(Q_{2B}), P_{T\,final}\right\}.\end{cases} \tag{18}$$

Since Q_1 is supposed to be optimal on the contractions of T_A and T_B, there are $P_T(Q_{1A}) \le P_T(Q_{2A})$ and $P_T(Q_{1B}) \le P_T(Q_{2B})$. Therefore, we have

$$\begin{aligned}&\max\left\{P_T(Q_{1A}), P_T(Q_{1B}), P_{T\,final}\right\}\\ &\le \max\left\{P_T(Q_{2A}), P_T(Q_{2B}), P_{T\,final}\right\},\end{aligned} \tag{19}$$

which is a contradiction. Therefore, the lemma is proved. ■

Lemma 3 *Consider an arbitrary tensor tree network T whose edges have the same weights, and Q_T to be an arbitrary sequence of contractions. For any adjacent two steps θ_k and θ_{k+1}, if we interchange* *them and thus construct a new sequence $Q_T^{'}$, we have*

$$P_T\left(Q_T\right) = P_T\left(Q_T^{'}\right). \tag{20}$$

Proof. We name the two edges whose contractions are interchanged as E_{A-B} and E_{C-D}. If $\{A, B\} \cap \{C, D\} = \emptyset$, their contractions are independent of each other, and thus can be interchanged. Otherwise, we suppose $A = D$ and turn to consider the contractions of E_{A-B} and E_{A-C}. Since these two steps are adjacent on the sequence, whether these two steps interchange will not affect the other steps.

If we contract A with B firstly and contract AB with C secondly, then the time power of the two steps as a whole is

$$\begin{aligned}&\max\{S_{WD}(A) + S_{WD}(B) - W_{A-B},\\ &S_{WD}(AB) + S_{WD}(C) - W_{A-C}\}\\ =&\max\{S_{WD}(A) + S_{WD}(B) - W_{A-B},\\ &S_{WD}(A) + S_{WD}(B) - 2\times W_{A-B} + S_{WD}(C) - W_{A-C}\}.\end{aligned} \tag{21}$$

As assumed, W_{A-B} and W_{A-C} are minimal. Therefore, if C is a leaf, then we have

$$S_{WD}(C) - W_{A-C} = W_C > W_{A-B}. \tag{22}$$

If C is not a leaf, then there exists another vertex E connected to C. In this case, there is

$$\begin{aligned}&S_{WD}(C) - W_{A-C} = W_C + \mathcal{D}(C) - W_{A-C}\\ \geq& W_C + W_{A-C} + W_{C-E} - W_{A-C}\\ =& W_C + W_{C-E}\\ =& W_C + W_{A-B}\\ \geq& W_{A-B}.\end{aligned} \tag{23}$$

Therefore, no matter whether C is a leaf or not, we always have

$$\begin{aligned}&S_{WD}(A) + S_{WD}(B) - 2\times W_{A-B} + S_{WD}(C) - W_{A-C}\\ >& S_{WD}(A) + S_{WD}(B) - 2\times W_{A-B} + W_{A-B}\\ =& S_{WD}(A) + S_{WD}(B) - W_{A-B}.\end{aligned} \tag{24}$$

Thus, we can get

$$\begin{aligned}&\max\{S_{WD}(A) + S_{WD}(B) - W_{A-B},\\ &S_{WD}(AB) + S_{WD}(C) - W_{A-C}\}\\ =& S_{WD}(A) + S_{WD}(B) - 2\times W_{A-B} + S_{WD}(C) - W_{A-C}.\end{aligned} \tag{25}$$

Similarly, if we contract A with C firstly followed by contracting AC with B, then the time power is

$$\begin{aligned}&\max\{S_{WD}(A) + S_{WD}(C) - W_{A-C},\\ &S_{WD}(AC) + S_{WD}(B) - W_{A-B}\}\\ =& S_{WD}(A) + S_{WD}(C) - 2\times W_{A-C} + S_{WD}(B) - W_{A-B}.\end{aligned} \tag{26}$$

Since $W_{A-B} = W_{A-C}$, we know that

$$\begin{aligned}&S_{WD}(A) + S_{WD}(B) - 2\times W_{A-B} + S_{WD}(C) - W_{A-C}\\ =& S_{WD}(A) + S_{WD}(C) - 2\times W_{A-C} + S_{WD}(B) - W_{A-B},\end{aligned} \tag{27}$$

which is to say that the two sequences have the same time power. Therefore, they can be interchanged. ■

Theorem 1 (Minimal Time Power) *For any tensor tree network T, suppose the sequence achieved by the algorithm is $Q_{T_{alg}}$. This leads to the following equation*

$$P_T\left(Q_{T_{alg}}\right) = \min_{Q_T}\{P_T\left(Q_T\right)\}. \tag{28}$$

Proof. We prove the theorem by mathematical induction.

Step 1: When $n = 1, 2$, the conclusion is trivial.

Step 2: When $n = 3$, without losing generality, we suppose the structure of the tree can be expressed as FIG. 6.

Therefore, if we firstly contract A with B, the time power is

$$\begin{aligned}\max\{&W_A + W_B + W_{A-B} + W_{B-C},\\ &W_A + W_B + W_C + W_{B-C}\}.\end{aligned} \tag{29}$$

If we firstly contract B with C, the time power is

$$\begin{aligned}\max\{&W_B + W_C + W_{A-B} + W_{B-C},\\ &W_A + W_B + W_C + W_{A-B}\}.\end{aligned} \tag{30}$$

Corresponding to the algorithm, we divide the condition into two cases respectively:

Case 2.1: If the minimum exists on leaves, without losing generality, we suppose the minimum is W_A. In this case, we have

$$\begin{aligned}W_A + W_B + W_{A-B} + W_{B-C} &\leq W_B + W_C + W_{A-B} + W_{B-C}\\ W_A + W_B + W_C + W_{B-C} &\leq W_B + W_C + W_{A-B} + W_{B-C}.\end{aligned} \tag{31}$$

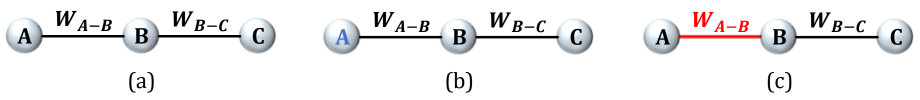

FIG. 6. A tensor tree network with n vertices when n=3 in (a). In specific, the case when $W_A = \min\{W_A, W_C, W_{A-B}, W_{B-C}\}$ and the case when $W_{A-B} = \min\{W_A, W_C, W_{A-B}, W_{B-C}\}$ are provided in (b) and (c), respectively.

Therefore, it comes the following inequation

$$\begin{aligned} &\max\{W_A + W_B + W_{A-B} + W_{B-C},\\ &\quad W_A + W_B + W_C + W_{B-C}\}\\ \leq\, & W_B + W_C + W_{A-B} + W_{B-C} \\ \leq\, & \max\{W_B + W_C + W_{A-B} + W_{B-C},\\ &\quad W_A + W_B + W_C + W_{A-B}\}. \end{aligned} \tag{32}$$

According to equations (29) and (30), it is better to contract A with B on time power.

Case 2.2: If the minimum does not exist on leaves, then the minimum must exist on an edge. Without losing generality, we suppose the minimum to be W_{A-B}. Therefore, we have

$$\begin{cases} W_A > W_{A-B},\\ W_C > W_{A-B},\\ W_{B-C} \geq W_{A-B}. \end{cases} \tag{33}$$

Furthermore, there are inequations

$$\begin{cases} W_A + W_B + W_C + W_{B-C} > W_A + W_B + W_{A-B} + W_{B-C},\\ W_A + W_B + W_C + W_{B-C} > W_B + W_C + W_{A-B} + W_{B-C}. \end{cases} \tag{34}$$

Therefore, we know

$$\begin{aligned} &\max\{W_A + W_B + W_{A-B} + W_{B-C},\\ &\quad W_A + W_B + W_C + W_{B-C}\}\\ =\, & W_A + W_B + W_C + W_{B-C} \\ \geq\, & \max\{W_B + W_C + W_{A-B} + W_{B-C},\\ &\quad W_A + W_B + W_C + W_{A-B}\}. \end{aligned} \tag{35}$$

According to equations (29) and (30), we can contract B with C firstly in order to achieve the minimal time power.

Note that our algorithm is proposed based on these two strategies and thus it can achieve the minimal time power when $n = 3$.

Step 3: Suppose the conclusion is true when $n \leq k$, where $k \geq 2$. Now we consider the case when $n = k + 1$.

Accordingly, we also divided the condition into 2 cases:

Case 3.1: If the minimum exists on one leaf, we give this leaf a name "A". Also, we suppose its weight to be W_A. Since A is a leaf, we denote its only adjacent vertex, which is also its father, as B with its weight W_B. The edge connecting A and B has its weight W_{A-B}.

Now we consider the contractions, which have the following three possibilities: i) B contracts with vertices other than A; ii) B contracts with A; iii) Contractions between other vertices.

On one hand, the time power of contracting B with A would not exceed that of contracting B with any other vertex C. In fact according to equation (12), the time power of contracting B with A, and B with C are sequentially

$$\begin{cases} S_{WD}(B) + S_{WD}(A) - W_{A-B},\\ S_{WD}(B) + S_{WD}(C) - W_{B-C}. \end{cases} \tag{36}$$

Since W_A is the least among weights of leaves and edges, and that $S_{WD}(A) = W_A + W_{A-B}$, we have the following considerations: i) If C is a leaf, $S_{WD}(C) = W_C + W_{B-C}$, then we have

$$\begin{aligned} & S_{WD}(B) + S_{WD}(A) - W_{A-B}\\ =\, & S_{WD}(B) + W_A\\ \leq\, & S_{WD}(B) + W_C\\ =\, & S_{WD}(B) + S_{WD}(C) - W_{B-C}; \end{aligned} \tag{37}$$

ii) If C is not a leaf, then there exists a vertex D connecting to C other than B. Therefore, we have

$$\begin{aligned} S_{WD}(C) =\, & W_C + \mathcal{D}(C)\\ \geq\, & W_C + W_{B-C} + W_{C-D}\\ \geq\, & 0 + W_{B-C} + W_A\\ =\, & W_{B-C} + W_A, \end{aligned} \tag{38}$$

and further have

$$\begin{aligned}&S_{WD}(B) + S_{WD}(A) - W_{A-B}\\ =&S_{WD}(B) + W_A\\ \leq&S_{WD}(B) + S_{WD}(C) - W_{B-C}.\end{aligned} \tag{39}$$

On the other hand, since $W_{B-C} = W_{AB-C}$, we know

$$\begin{aligned}&S_{WD}(B) + S_{WD}(C) - W_{B-C}\\ =&S_{WD}(B) + S_{WD}(C) - W_{AB-C}\\ \geq&S_{WD}(B) + S_{WD}(A) - 2W_{A-B} + S_{WD}(C) - W_{AB-C}\\ =&S_{WD}(AB) + S_{WD}(C) - W_{AB-C}.\end{aligned} \tag{40}$$

Therefore, the time power of contracting AB with any other vertex C would not exceed that of contracting B with C.

With the two facts above, we consider the following case: if we contract B with C just before A being contracted, we can firstly contract A with B and secondly contract AB with C because this would not make the time power increase. As a result, without losing generality, we can firstly contract the leaf A, whose weight is the minimum among the weights of edges and leaves, with its father vertex. Thus, this case is reduced to the case that $n \leq k$.

Case 3.2: If the minimum does not exist on leaves, then the minimum must exist on an edge and the weight of every leaf must be strictly greater than the minimal edge weight. According to our algorithm, we should find out the edge with a minimal weight, and stipulate this edge to be eliminated in the last (or named the "k^{th}") step. After that, this edge divides the tree into two subtrees. We then add the weights of the two endings of this edge by the minimal weight for each, in their subtrees, and deal with the two subtrees independently. Now we prove that the above strategy can achieve the minimal time power.

In the case that $n = k + 1$, we prove the result by contradiction. Suppose there exist sequences of eliminating edges that can achieve the minimal time power, and the edge contracted in the last step is never the one with the minimal weight. Now we select one sequence from those optimal sequences, and consider this sequence in the following part. If the edge contracted in the last step is E_{A-B}, then this edge divides the tree into two subtrees, and each of them has a root A or B. As a result, the edge with a minimal weight must exist at least in one of the two subtrees.

Therefore, contractions before the last step can be seen as those in the two subtrees, independently. If we consider one of the subtrees individually, we should take into consideration the edge to be contracted during the last step. Consequently, while considering the subtree whose root is A, we should give A a new weight $W_A' = W_A + W_{A-B}$, in order to keep $S_{WD}(A)$ unchanged. This operation is similar to B, that is to say, $W_B' = W_B + W_{A-B}$. For the purpose of expression, we name the two subtrees as T_A and T_B.

Note that W_{A-B} is not the least, and thus neither A nor B will become a leaf whose weight is less than the least weight of an edge. Now we go back to the original tree, find out an edge with the minimal weight, and name it as E_{C-D}. Since the division of A and B does not damage other edges, E_{C-D} exists in either T_A or T_B. Without losing generality, we suppose that E_{C-D} exists in T_A.

We know that the result of contracting T_A and T_B is not relevant to the sequence of contracting them. According to Lemma 2, for an optimal sequence along with the edge being eliminated in the final step ($A - B$), we can guarantee the sequence to be optimal by contracting T_A and T_B optimally. Since the number of vertices in T_A or T_B is less than k, we can optimally contract T_A and T_B in our algorithm. As is supposed that E_{C-D} exists in T_A, the last step in the optimal sequence of contractions of T_A is to eliminate E_{C-D}. Without losing generality, we assume that D is nearer to A than C (actually, D might be exactly A).

Similar to the nomination of T_A and T_B, we name the two subtrees of T_A, divided by edge E_{C-D}, as T_C and T_{AD}, which respectively represent the subtree with root C and the subtree between A and D. Also, the contractions in T_C, T_{AD} and T_B are independent of each other, and according to Lemma 2, we can optimally contract them individually.

Now we consider the last two steps of finding the optimal sequence. Currently, there are three vertices and 2 edges E_{A-B} and E_{C-D} remained. On one hand, it has been supposed at the beginning that the time power to contract E_{A-B} after contracting E_{C-D} is optimal, and it is strictly less than that of contracting an edge with a minimal weight, such as E_{C-D}, in the last step. However, on the other hand, as is proved in the case $n = 3$, an optimal sequence is to contract E_{C-D} in the final step. This is because that, since W_{C-D} is strictly less than the weight of any leaf and the weight of any newly born leaf would not be less than that of existed ones, W_{C-D} is always the minimal weight among those of leaves and edges. According to our conclusion in the case $n = 3$, we contract E_{C-D} in the last step to achieve the optimal sequence.

Therefore, our earlier assumption contradicts to what we have proved in the case $n = 3$. Thus, in the case that $n = k + 1$ and the minimal weight does not exist on a leaf, we can achieve a sequence of optimal time power by contracting (eliminating) the edge with a minimal weight in the last step.

So far, we have only proved the necessity, and now consider the sufficiency. If there is only one minimally weighted edge, the sufficiency is trivial. For the case that there is more than one, we know that they are contracted after those edges whose weights are larger than these minima. According to Lemma 3, the sequence of contracting these minimally weighted edges can be interchanged, and thus can be arbitrarily arranged. Therefore, the sequence of contracting them does not affect the time power of the whole process. This proves our algorithm to be sufficient.

In conclusion, we have proved that our algorithm is valid in the case $n = k + 1$, and thus our algorithm can achieve the minimal time power for any tree graph module, according to the principle of induction. ■

B. Optimal Space Complexity

Theorem 2 (Minimal Space Power) *For a tree T, we denote its final result of contractions, which is a single vertex, as T_c. By applying the algorithm proposed in Section IV to contract T, the space power for every step will not exceed*

$$\max\left\{\max_{A\in T}\{S_{WD}(A)\}, W_{T_c}\right\}. \tag{41}$$

Proof. Firstly, we have mentioned while dividing a tree into two subtrees from one edge each time, the two subtrees are not achieved directly. Instead, for each subtree, before we consider it as a tree, we should conduct an operation: find out the end vertex of the "split" edge, add the weight of this "split" edge of this vertex. Therefore, the S_{WD} values of these two end vertices keep unchanged in the original tree and in the subtrees. As a result, every vertex maintains its S_{WD} value in the original tree and subtrees. Therefore, we only need to consider the influence of each contraction (or elimination) on the subtree it belongs to, instead of considering the influence on the whole tree.

Now we consider the two operations adopted during contraction. *Operation 1*: If the minimal weight of leaves and edges emerges on a leaf, then we contract the leaf to its father. Apparently, during this contraction, the S_{WD} of the father vertex is larger than that of the leaf. Also, the combined vertex born in contraction has a lower S_{WD} value in comparison with that of the father vertex. Therefore, this contraction does not emerge new vertices whose S_{WD} exceeds the maximum among already existing ones. *Operation 2*: If the minimal weight of leaves and edges does not emerge on a leaf, then it must emerge on an edge. Therefore, we must eliminate an edge with the minimal weight in the final step of contraction. This result in that the sum of the weights of the two remaining vertices equals the sum of the weights of all vertices in the tree (subtree). Thus the sum cannot exceed the sum of vertices in the tree.

As there are only these two cases, we have proved the theorem. ■

Remark 4. Since the terms in equation (41) exist definitely, this expression is exactly the optimal space power. In other words, it is the greatest lower bound and this space power is the best result among all kinds of contractions, since this result measures the necessary space complexity for the storage of inputs and outputs. Therefore, in the following proof of edge contractions to be a best stipulation, we will only prove the optimization on time power.

C. Edge Contractions

In this subsection, we prove our stipulation of "edge contraction", which forbids the contractions between two vertices that are not connected, as formally stated in the following theorem.

Theorem 3 (Edge Contraction) *For an arbitrarily connected tensor network and any sequence of contraction Q_1, if there are two non-connected vertices contracted in Q_1, then there must exist a different sequence of contraction Q_2 such that: (1) No non-connected vertices are contracted in Q_2 and (2) $P_T(Q_1) \geq P_T(Q_2)$.*

Proof. The result is proved by contradiction. For the steps that contract two non-connected vertices each time in Q_1, we consider the last one of these non-connected-contraction steps. We denote the two vertices as A and B and the combined (result-of-contraction) vertex as AB. It is apparent that a connected network would not be changed to a non-connected one after any step of contraction. As a result, we know this non-connected contraction is not the final step of the whole sequence of contractions. Therefore, AB will be contracted with another vertex in one of the future steps of contractions, and we denote this vertex as C. On one hand, for contractions between any vertices other than A or B, the time power of these contractions would not be affected by whether A or B is contracted or not. On the other hand, the time power of contracting AB with C is

$$\begin{aligned} &S_{WD}(AB) + S_{WD}(C) - W_{AB-C} \\ =&S_{WD}(A) + S_{WD}(B) + S_{WD}(C) - W_{A-C} - W_{B-C} \\ =&(S_{WD}(A) - W_{A-C}) + (S_{WD}(B) - W_{B-C}) + S_{WD}(C). \end{aligned} \tag{42}$$

Since the contraction of A with B is the last step of non-connected contractions, the contraction of AB with C must be an edge contraction. Therefore, C must connect with at least one of A and B. Without losing generality, we suppose A and C are connected. If we firstly contract A and C as AC, then contract B with AC, the result is the same and thus the time power of contractions after these two steps remains unchanged. However, the time power of these two steps of contractions turns out to be

$$\begin{aligned} \max\{&S_{WD}(A) + S_{WD}(C) - W_{A-C}, \\ &S_{WD}(B) + S_{WD}(AC) - W_{B-AC}\}. \end{aligned} \tag{43}$$

Since we have

$$\begin{aligned} S_{WD}(AC) &= S_{WD}(A) + S_{WD}(C) - 2 \times W_{A-C} \\ W_{B-AC} &= W_{A-B} + W_{B-C} = W_{B-C}, \end{aligned} \tag{44}$$

the new time power equals

$$\begin{aligned} &\max\{S_{WD}(A) + S_{WD}(C) - W_{A-C}, \\ &S_{WD}(B) + S_{WD}(A) + S_{WD}(C) - 2 \times W_{A-C} - W_{B-C}\} \\ =&\max\{(S_{WD}(A) - W_{A-C}) + S_{WD}(C), \\ &(S_{WD}(B) - W_{B-C}) + (S_{WD}(A) - W_{A-C}) \\ &+ (S_{WD}(C) - W_{A-C})\}. \end{aligned} \tag{45}$$

Notice the inequation

$$\begin{aligned} &(S_{WD}(A) - W_{A-C}) + (S_{WD}(B) - W_{B-C}) + S_{WD}(C) \\ \geq&(S_{WD}(A) - W_{A-C}) + S_{WD}(C), \end{aligned} \tag{46}$$

and

$$\begin{aligned} &(S_{WD}(A) - W_{A-C}) + (S_{WD}(B) - W_{B-C}) + S_{WD}(C) \\ \geq&(S_{WD}(B) - W_{B-C}) + (S_{WD}(A) - W_{A-C}) \\ &+ (S_{WD}(C) - W_{A-C}). \end{aligned} \tag{47}$$

Therefore, the inequation comes out as

$$\begin{aligned} &(S_{WD}(A) - W_{A-C}) + (S_{WD}(B) - W_{B-C}) + S_{WD}(C) \\ \geq&\max\{(S_{WD}(A) - W_{A-C}) + S_{WD}(C), (S_{WD}(B) - W_{B-C}) \\ &+ (S_{WD}(A) - W_{A-C}) + (S_{WD}(C) - W_{A-C})\}. \end{aligned} \tag{48}$$

With this rearrangement, the time power would at least remain unchanged, if not decrease.

Also, if B and C are connected with a non-zero-weighted edge, then the contraction of AC and B is also an edge contraction, and thus the total number of non-connected contraction decreases by one. Otherwise, the contraction between B and AC is a non-connected contraction, which means that the last step of non-connected contractions moves later by one step. Since there are all edge contractions after the step of contracting B and AC, this step is the current last step of non-connected contractions, and we can consider this step in a similar way as we have just conducted. According to equation (48), an adjustment on the sequence can either change this non-connected contraction to an edge contraction, or move later the last step of non-connected contractions by one step. Since the total number of contractions is limited, and the fact that the final step of the whole contractions must be an edge contraction, we know that the selection of moving later by one step is not always valid. Thus, there must

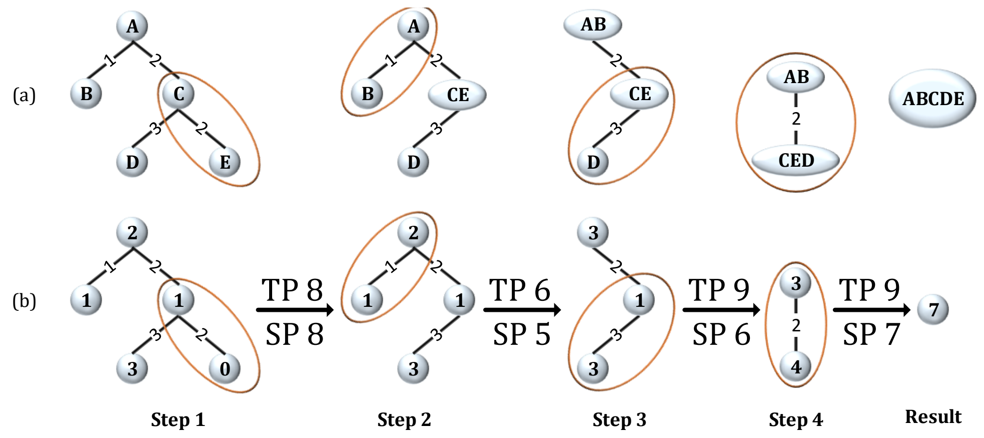

FIG. 7. The optimal sequence of contracting a tensor tree network: (a) The sequence our algorithm determines; (b) Time power and space power for each step. For the optimal sequence shown here, the time power for the whole process is 9, and the space power is 8.

exist a time when the rearrangement results in an edge contraction, and the number of non-connected contractions will be reduced by one. In conclusion, we can always reduce the number of non-connected contractions by one in limited operations of adjustments on the sequences. As a result, the number of non-connected contractions can be reduced to zero after our finite number of operations of adjustments. In other words, there exists a sequence, achieved by adjusting from our original one, that has no non-connected contractions and has a time power which is no more than that of our original sequence. Thus we have proved this theorem. ■

D. Polynomial Complexity for Algorithm Execution

In the three subsections above we have proved the validity of our algorithm. Now we will prove that it is a polynomial algorithm.

Theorem 4 (Polynomial Algorithm) *The algorithm proposed in Section IV can be executed in $O\left(n^2\right)$ time complexity.*

Proof. For every loop in executing our algorithm, we only need to find out the minimal element in the set of weights of edges and leaves, whose number of elements is at most $2n$. It takes $O(n)$ time complexity to find out the minimal element from a set of $2n$ elements. Also, after executing every loop in the algorithm, we can select one edge to be connected in either the first or the last step among the following steps of contractions, and thus the algorithm can be executed in $n-1$ steps. Therefore, the total time complexity of the algorithm is $O(n^2)$. ■

V. EXAMPLE AND SIMULATIONS

In this section, an example and some numerical simulations of contracting tensor tree networks are demonstrated.

A. An Example of Tree Network Contraction

We firstly give an example to illustrate our algorithm, as shown in FIG. 7 where we calculate the following expression:

$$\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N}\sum_{p=1}^{N}\sum_{q=1}^{N}\sum_{r=1}^{N}\sum_{s=1}^{N}\sum_{t=1}^{N} A_{i_1,i_2,j,k,l}\cdot B_{i_3,j} \cdot C_{i_4,k,l,p,q,r,s,t}\cdot D_{i_5,i_6,i_7,p,q,r}\cdot E_{s,t}. \quad (49)$$

TABLE I. Stipulations and Sequences

Parameters	Stipulations or Sequences
Weights of leaves	0 (65% possibility) or 1 (35% possibility)
Weights of other vertices	0
Weights of edges	Among {1,2,3} randomly
Number of layers	From 2 to 9 sequentially
Number of vertices in each layer	Among {1,2,3} randomly
N	From 2 to 5 sequentially

According to the proposed algorithm, the minimal weight of edges and leaves belongs to the leaf E, whose weight is zero. Therefore, we contract E with its father C firstly, to be CE, which is not a leaf. Secondly, the minimal weight belongs to the leaf B, whose weight is one, and we contract B with its father A, to be AB. After that, the minimal weight belongs to the edge E_{AB-CE}, and thus we contract (eliminate) this edge in the last step. Therefore, we contract E_{CE-D} prior to E_{AB-CE}. The time power emerges in the third and the fourth steps, and the time power of this sequence of contractions equals 9. The space power emerges in the original vertex C. This example would help to understand our algorithm.

B. Simulations on Tensor Network Contractions

Now we carry out numerical simulations to illustrate the efficiency of our algorithm on large numbers of random tensor tree networks. We compare the time power and the total number of multiplications of applying our algorithm with that of applying a random sequence of contraction for every randomly-structured tree in different conditions.

The following main factors that affect the time and space complexity, or cost, are considered while conducting simulations. The first is the number of orders of tensors (the $S_{WD}(\cdot)$), including the weight of each vertex and each edge. Another is the dimension of each order of tensors, which is denoted as N in the sections above. Besides, the size of the tree network also plays an important role, including the depth (the number of layers in a tree once we select a root) and the width (the number of vertices in each layer).

Our goal is to testify the validity of our algorithm in all kinds of tensor tree networks and to illustrate the advantages of our algorithm over a random sequence of contraction. For the former one, the selection of tree network must be random. However, considering the computational bound of our device, we stipulate that the weight of a leaf is either 0 (65% probability) or 1 (35% probability), and the weights of edges range from 1 to 3 randomly. Also, we stipulate that the tree width ranges from 1 to 3 randomly. These stipulations are applied in every simulation. For the latter one, we would like to verify that our algorithm can save more time or space as the scale of the problem increases. Therefore, we carried out the simulation with the following sequences: the depths of the tree range from 2 to 9 sequentially, and N ranges from 2 to 5 sequentially. These stipulations and sequences are summarized in Table I.

In order to evaluate the efficiency of our algorithm in practical scenarios, we develop the following 4 indicators for every random tree: (1) Running time, including the time for contraction and the time for executing our algorithm; (2) Time for contraction; (3) Storage space; (4) Computational cost that counts the total amount of multiplications.

For each configuration of the tree depth and N, we set up 500 tensor tree networks randomly, according to our stipulations. We execute our algorithm on each of them and calculate the average of the 500 results on each indicator individually. We then plot these averages according to the sequence on charts, as in FIG. 8. To clearly show the advantages of our algorithm over random sequences, which can be excessively large, we use logarithmic coordinates for some indicators.

From FIG. 8 we have noticed the following ob-

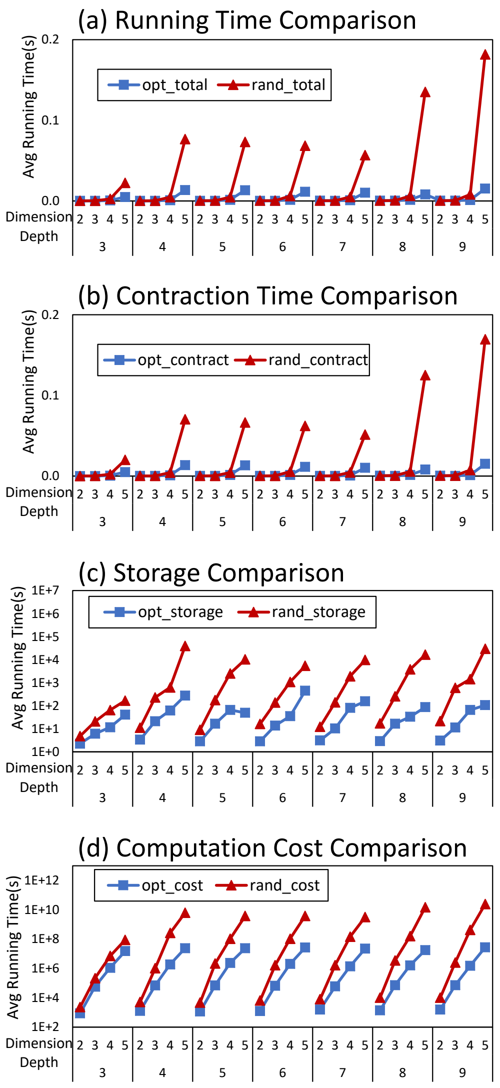

FIG. 8. These four charts compare the numerical simulation properties of our algorithm with that of random contractions, from different perspectives: (a) The total time, including the time spent on our algorithm, the procedure of contraction and rearrangement of elements; (b) Time on contractions, with each point on these two plots representing an average of 100 random cases; (c) and (d) are logarithmic algorithms with logs base 10, which sequentially express the storage(space) cost and computational (numbers of multiplications) cost. The results show the advantages of our algorithm over a random sequence of contractions on these four scales.

servations:

1. Overall, the results of optimal sequences of contraction achieved via our algorithm are always better than those of the random ones in any condition and under all of these indicators.

2. With the increase of tree depth, both optimal and random sequences are more time-and-space-consuming. Our algorithm presents an increasing advantage on every indicator: it spends $1/10^3$ of computational cost and storage on the condition when $Depth = 9$, but it expresses imperceptible advantages when $Depth = 2$ (both under the condition $N = 5$). For the running and contraction time, our algorithm also shows increasing benefits.

3. With the increase of dimension N of each order of tensors, the problem is apparently more exhausting. Actually, the advantages of our algorithm become more significant on any indicator as N increases.

C. Simulations on Searching for Sequences

Now we present another set of numerical experiments. Since our algorithm is proved to be a polynomial-time algorithm, we conjecture that the algorithm presented in [29] should be more time consuming than ours while performing on the same tensor tree networks. Therefore, we have carried out the following numerical simulations, comparing these two algorithms in terms of execution time. We use the MATLAB code from [29] and a Python code for our algorithm. All inputs are constructed similarly to those in the last subsection V B. For simplicity, we set the number of vertices in each layer as a fixed number 3 in all tests, and change only the tree depth to reflect the increase of scale. In this work, we only aim to find the contraction sequence of a network rather than doing real tensor contractions. Thus, the value of N and the distribution of weights are not necessary. In other words, we just need to guarantee that all input weights are no more than the same constant number.

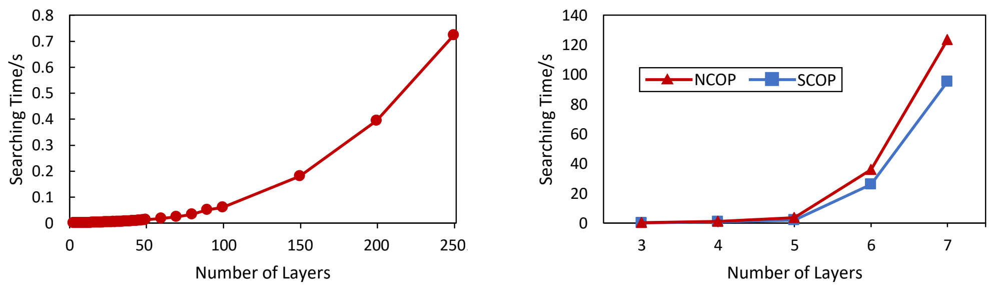

FIG. 9. Comparison of the execution time between our algorithm and the one in [29]. We construct a tree by starting from a root and giving each layer 3 vertices, and then we randomly connect vertices between adjacent layers without forming any loop. Weights of vertices and edges are randomly chosen non-negative real numbers within a constant range. Specifically, (a) we plot the execution time of our algorithm on trees with different number of layers, and (b) plot the search time of the algorithm [29] on trees with different number of layers. "NCOP" and "SCOP" in (b) represent "Not Consider Outer Product" and "Should Consider Outer Product", respectively, due to an alternate trade-off of their algorithm. It is apparent that our algorithm can perform much better than the prior one.

Fig. 9 depicts the simulation results. Fig. 9(a) reveals that our algorithm can find an optimal sequence within one second on average even on trees with 250 layers and 751 vertices totally. Further analysis shows that the trend of this series of time could be well represented by a polynomial within $O(n^3)$, which is exactly the time complexity of our code and could still be optimized until $O(n^2)$. In comparison, Fig. 9(b) shows that the algorithm presented in [29] is much more time-consuming. As is shown, performing on a network with only 7 layers would take more than 100s. If the number of layer exceeds 8, our personal computer would no longer be able to execute their algorithm due to the insufficient memory space. Apparently, all those huge, and even increasing, gaps could not be just owed to the difference between Python and MATLAB. In a nutshell, our algorithm is more efficient in finding an optimal sequence of contracting tensor tree networks.

To be honest, the algorithm in [29] have already saved a large number of unnecessary searches (e.g. remove the search of outer products) and thus saved a large amount of time, but it is still not a polynomial one. Their dynamic-programming based algorithm would be much less efficient than ours, even if in the tree cases where their algorithm can perform better than other networks due to the more removable unnecessary searches.

VI. CONCLUSION AND DISCUSSION

In this paper, we first transform the NP-Complete problem on minimizing the computational cost of contracting a general tensor network to an open problem on minimizing the time and space complexities. This kind of transformation leads to a simplification of calculation, as it originally needs additions and multiplications when calculating a cost but now only additions (on the index of power) when calculating the complexity. Moreover, our work contributes to the efficiency of tensor network contraction, by figuring out the optimal sequence of contracting an important tensor network structure, the tensor tree network. We propose a polynomial algorithm to find the optimal contraction sequence on tensor tree networks, and we prove that this algorithm indeed achieves the optimal time and space complexity. In order to illustrate the effectiveness of our algorithm, we also carry out numerical simulations of tensor tree network contractions and compare the time and storage cost of the contraction sequences with those of random sequences, respectively.

In addition, we believe that the solution of tree networks would serve as a key to that of more complex networks, or even arbitrary ones. For a cycle with n vertices, we can first enumerate all possible operations of the final step and then apply our algorithm on the previous part, which has been "split" into two chains (also trees in general) by the final step. In this way, the time complexity to achieve the optimal contraction sequence of an n-vertex cycle is $O(n^4)$ in total, revealing that this subproblem is polynomial. Similar polynomial methods can be applied on any tensor network with a constant number of loops. Also, some important properties of the tensor tree network, such as the edge with a minimal weight, might be taken into consideration in our future research on general networks. Besides of those inspiring properties, tensor tree network itself has plenty of applications involving simulations of physics systems, as aforementioned in Section I. These important applications would reflect the significance of our work to a great extent.

However, there are still many issues remaining to be addressed. First of all, the problem of achieving the optimal computational complexity of contracting an arbitrary complete graph has not been classified into any complexity class, though the problem of achieving the optimal total computational cost has been proved to be NP-Complete. This is equivalent to the question whether our transformation from "cost" to "complexity" is really a "simplification" or not, because the hardness of these two problem would be identical if the problem "complexity" is also proved to be NP-complete. Second, it is unknown whether the optimal time complexity can always come up simultaneously with an optimal space complexity in a complete graph, although it has been achieved in our algorithm on the tensor tree network. Furthermore, although we have proved that our algorithm can achieve an optimal sequence of contractions, it is not clear whether the method of contractions is efficient enough for calculating tensor networks or not. For instance, the time complexity of computing a matrix-chain product is $O(N^3)$ according to our algorithm based on the framework of tensor contractions, but there do exist methods that calculate matrix products differently and can guarantee a lower complexity [53]. These problems are attractive but challenging, which deserve further researches.

ACKNOWLEDGMENT

This work was partially supported by National Natural Science Foundation of China (Grant No. 61876215), National Science Foundation (Grant No. 1817037), Tsinghua University Initiative Scientific Research Program, and Tsinghua-Foshan Industry-University-Research Cooperation and Innovation Funds. We also gratefully acknowledge Mr. Pengfei Yu, Mr. Xuanyuan Luo and Mr. Zhiyu Wei, all from Tsinghua University, for their salutary attempts and precious inspirations on the design of our algorithm.

[1] Román Orús, "A practical introduction to tensor networks: Matrix product states and projected entangled pair states," Annals of Physics **349**, 117–158 (2014).

[2] Jacob Biamonte and Ville Bergholm, "Quantum tensor networks in a nutshell," arXiv preprint arXiv:1708.00006 (2017).

[3] Andrzej Cichocki, "Tensor networks for big data analytics and large-scale optimization problems," arXiv preprint arXiv:1407.3124 (2014).

[4] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Advances in neural information processing systems* (2013) pp. 926–934.

[5] Cupjin Huang, Michael Newman, and Mario Szegedy, "Explicit lower bounds on strong quantum simulation," arXiv preprint arXiv:1804.10368 (2018).

[6] Frank Verstraete and J Ignacio Cirac, "Renormalization algorithms for quantum-many body systems in two and higher dimensions," arXiv preprint cond-mat/0407066 (2004).

[7] Frank Verstraete, Valentin Murg, and J Ignacio Cirac, "Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems," Advances in Physics **57**, 143–224 (2008).

[8] Andreas Holzner, Andreas Weichselbaum, and Jan von Delft, "Matrix product state approach for a two-lead multilevel anderson impurity model," Physical Review B **81**, 125126 (2010).

[9] David Perez-Garcia, Frank Verstraete, Michael M Wolf, and J Ignacio Cirac, "Matrix product state representations," arXiv preprint quant-ph/0608197 (2006).

[10] Robert Hübener, Volckmar Nebendahl, and Wolfgang Dür, "Concatenated tensor network states," New Journal of Physics **12**, 025004 (2010).

[11] G Scarpa, A Molnar, Y Ge, JJ Garcia-Ripoll, N Schuch, D Perez-Garcia, and S Iblisdir, "Computational complexity of peps zero testing," arXiv preprint arXiv:1802.08214 (2018).

[12] Andras Molnar, José Garre-Rubio, David Pérez-García, Norbert Schuch, and J Ignacio Cirac, "Normal projected entangled pair states generating the same state," arXiv preprint arXiv:1804.04964 (2018).

[13] Michael Lubasch, J Ignacio Cirac, and Mari-Carmen Banuls, "Unifying projected entangled pair state contractions," New Journal of Physics **16**, 033014 (2014).

[14] Szilárd Szalay, Max Pfeffer, Valentin Murg, Gergely Barcza, Frank Verstraete, Reinhold Schneider, and Örs Legeza, "Tensor product methods and entanglement optimization for ab initio quantum chemistry," International Journal of Quantum Chemistry **115**, 1342–1391 (2015).

[15] Philippe Corboz and Guifré Vidal, "Fermionic multiscale entanglement renormalization ansatz," Physical Review B **80**, 165129 (2009).

[16] Takeshi Yanai, Haruyuki Nakano, Takahito Nakajima, Takao Tsuneda, So Hirata, Yukio Kawashima, Yoshihide Nakao, Muneaki Kamiya, Hideo Sekino, and Kimihiko Hirao, "Utchema program for ab initio quantum chemistry," in *International Conference on Computational Science* (Springer, 2003) pp. 84–95.

[17] Sergio Boixo, Sergei V Isakov, Vadim N Smelyanskiy, and Hartmut Neven, "Simulation of low-depth quantum circuits as complex undirected graphical models," arXiv preprint arXiv:1712.05384 (2017).

[18] Jianxin Chen, Fang Zhang, Mingcheng Chen, Cupjin Huang, Michael Newman, and Yaoyun Shi, "Classical simulation of intermediate-size quantum circuits," arXiv preprint arXiv:1805.01450 (2018).

[19] Xipeng Qiu and Xuanjing Huang, "Convolutional neural tensor network architecture for community-based question answering." in *IJCAI* (2015) pp. 1305–1311.

[20] Dong Yu, Li Deng, and Frank Seide, "The deep tensor neural network with applications to large vocabulary speech recognition," IEEE Transactions on Audio, Speech, and Language Processing **21**, 388–396 (2013).

[21] Wenzhe Pei, Tao Ge, and Baobao Chang, "Max-margin tensor neural network for chinese word segmentation," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1 (2014) pp. 293–303.

[22] Yinchong Yang, Denis Krompass, and Volker Tresp, "Tensor-train recurrent neural networks for video classification," arXiv preprint arXiv:1707.01786 (2017).

[23] Timur Garipov, Dmitry Podoprikhin, Alexander Novikov, and Dmitry Vetrov, "Ultimate tensorization: compressing convolutional and fc layers alike," arXiv preprint arXiv:1611.03214 (2016).

[24] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov, "Tensorizing neural networks," in *Advances in Neural Information Processing Systems* (2015) pp. 442–450.

[25] Andrzej Cichocki, Danilo Mandic, Lieven De Lathauwer, Guoxu Zhou, Qibin Zhao, Cesar Caiafa, and Huy Anh Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," IEEE Signal Processing Magazine **32**, 145–163 (2015).

[26] Wolfgang Hackbusch, *Tensor spaces and numerical tensor calculus*, Vol. 42 (Springer Science & Business Media, 2012).

[27] Glen Evenbly and Robert NC Pfeifer, "Improving the efficiency of variational tensor network algorithms," Physical Review B **89**, 245118 (2014).

[28] Igor L Markov and Yaoyun Shi, "Simulating quantum computation by contracting tensor networks," SIAM Journal on Computing **38**, 963–981 (2008).

[29] Robert NC Pfeifer, Jutho Haegeman, and Frank Verstraete, "Faster identification of optimal contraction sequences for tensor networks," Physical Review E **90**, 033315 (2014).

[30] Robert NC Pfeifer, Glen Evenbly, Sukhwinder Singh, and Guifre Vidal, "Ncon: A tensor network contractor for matlab," arXiv preprint arXiv:1402.0939 (2014).

[31] Albert Hartono, Alexander Sibiryakov, Marcel Nooijen, Gerald Baumgartner, David E Bernholdt, So Hirata, Chi-Chung Lam, Russell M Pitzer, J Ramanujam, and P Sadayappan, "Automated operation minimization of tensor contraction expressions in electronic structure calculations," in *International Conference on Computational Science* (Springer,

2005) pp. 155–164.

[32] So Hirata, "Tensor contraction engine: Abstraction and automated parallel implementation of configuration-interaction, coupled-cluster, and many-body perturbation theories," The Journal of Physical Chemistry A **107**, 9887–9897 (2003).

[33] Qingda Lu, Xiaoyang Gao, Sriram Krishnamoorthy, Gerald Baumgartner, J Ramanujam, and Ponnuswamy Sadayappan, "Empirical performance model-driven data layout optimization and library call selection for tensor contraction expressions," Journal of Parallel and Distributed Computing **72**, 338–352 (2012).

[34] Albert Hartono, Qingda Lu, Xiaoyang Gao, Sriram Krishnamoorthy, Marcel Nooijen, Gerald Baumgartner, David E Bernholdt, Venkatesh Choppella, Russell M Pitzer, J Ramanujam, *et al.*, "Identifying cost-effective common subexpressions to reduce operation count in tensor contraction evaluations," in *International Conference on Computational Science* (Springer, 2006) pp. 267–275.

[35] Chi-Chung Lam, P Sadayappan, and Rephael Wenger, "On optimizing a class of multi-dimensional loops with reductions for parallel execution," Parallel Processing Letters **7**, 157–168 (1997).

[36] Wolfgang Hackbusch and Stefan Kühn, "A new scheme for the tensor representation," Journal of Fourier analysis and applications **15**, 706–722 (2009).

[37] Naoki Nakatani and Garnet Kin-Lic Chan, "Efficient tree tensor network states (ttns) for quantum chemistry: Generalizations of the density matrix renormalization group algorithm," The Journal of chemical physics **138**, 134113 (2013).

[38] Valentin Murg, Frank Verstraete, Reinhold Schneider, Péter R Nagy, and O Legeza, "Tree tensor network state with variable tensor order: An efficient multireference method for strongly correlated systems," Journal of chemical theory and computation **11**, 1027–1036 (2015).

[39] Boris N Khoromskij, "Tensors-structured numerical methods in scientific computing: Survey on recent advances," Chemometrics and Intelligent Laboratory Systems **110**, 1–19 (2012).

[40] Lars Grasedyck, Daniel Kressner, and Christine Tobler, "A literature survey of low-rank tensor approximation techniques," GAMM-Mitteilungen **36**, 53–78 (2013).

[41] Shi-Ju Ran, Emanuele Tirrito, Cheng Peng, Xi Chen, Gang Su, and Maciej Lewenstein, "Review of tensor network contraction approaches," arXiv preprint arXiv:1708.09213 (2017).

[42] Luca Tagliacozzo, Glen Evenbly, and Guifré Vidal, "Simulation of two-dimensional quantum systems using a tree tensor network that exploits the entropic area law," Physical Review B **80**, 235127 (2009).

[43] Y-Y Shi, L-M Duan, and Guifre Vidal, "Classical simulation of quantum many-body systems with a tree tensor network," Physical review a **74**, 022320 (2006).

[44] Pietro Silvi, Vittorio Giovannetti, Simone Montangero, Matteo Rizzi, J Ignacio Cirac, and Rosario Fazio, "Homogeneous binary trees as ground states of quantum critical hamiltonians," Physical Review A **81**, 062335 (2010).

[45] Wei Li, Jan von Delft, and Tao Xiang, "Efficient simulation of infinite tree tensor network states on the bethe lattice," Physical Review B **86**, 195137 (2012).

[46] J Ignacio Cirac and Frank Verstraete, "Renormalization and tensor product states in spin chains and lattices," Journal of Physics A: Mathematical and Theoretical **42**, 504004 (2009).

[47] Philippe Corboz, Glen Evenbly, Frank Verstraete, and Guifré Vidal, "Simulation of interacting fermions with entanglement renormalization," Physical Review A **81**, 010303 (2010).

[48] Örs Legeza, Thorsten Rohwedder, Reinhold Schneider, and Szilárd Szalay, "Tensor product approximation (dmrg) and coupled cluster method in quantum chemistry," in *Many-Electron Approaches in Physics, Chemistry and Mathematics* (Springer, 2014) pp. 53–76.

[49] Yong-Deok Kim and Seungjin Choi, "Nonnegative tucker decomposition," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on* (IEEE, 2007) pp. 1–8.

[50] Christian Lubich, Thorsten Rohwedder, Reinhold Schneider, and Bart Vandereycken, "Dynamical approximation by hierarchical tucker and tensor-train tensors," SIAM Journal on Matrix Analysis and Applications **34**, 470–494 (2013).

[51] Daniel Bauernfeind, Manuel Zingl, Robert Triebl, Markus Aichhorn, and Hans Gerd Evertz, "Fork tensor-product states: efficient multiorbital real-time dmft solver," Physical Review X **7**, 031013 (2017).

[52] Stefan Handschuh, "Changing the topology of tensor networks," arXiv preprint arXiv:1203.1503 (2012).

[53] Volker Strassen, "Gaussian elimination is not optimal," Numerische mathematik **13**, 354–356 (1969).