Computer Physics Communications 319 (2026) 109918

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Computational Physics

TTNOpt: Tree tensor network package for high-rank tensor compression

Ryo Watanabe [a,*], Hidetaka Manabe [a], Toshiya Hikihara [b], Hiroshi Ueda [c,d]

[a] Graduate School of Engineering Science, The University of Osaka, 1-3 Machikaneyama, Toyonaka, Osaka, 560-8531, Japan
[b] Graduate School of Science and Technology, Gunma University, Kiryu, Gunma, Japan
[c] Center for Quantum Information and Quantum Biology, The University of Osaka, Toyonaka, 560-0043, Japan
[d] Computational Materials Science Research Team, RIKEN Center for Computational Science (R-CCS), Kobe, Hyogo, 650-0047, Japan

A R T I C L E I N F O

Prof. Andrew Hazel

Keywords:
Tensor network method
Tree tensor network
Quantum spin system
Tensor decomposition
Data analysis

A B S T R A C T

We have developed TTNOpt, a software package that utilizes tree tensor networks (TTNs) for quantum spin systems and high-dimensional data analysis. TTNOpt provides efficient and powerful TTN computations by locally optimizing the network structure, guided by the entanglement pattern of the target tensors. For quantum spin systems, TTNOpt searches for the ground state of Hamiltonians with bilinear spin interactions and magnetic fields, and computes physical properties of these states, including the variational energy, bipartite entanglement entropy (EE), single-site expectation values, and two-site correlation functions. Additionally, TTNOpt can target the lowest-energy state within a specified subspace, provided that the Hamiltonian conserves total magnetization. For high-dimensional data analysis, TTNOpt factorizes complex tensors into TTN states that maximize fidelity to the original tensors by optimizing the tensors and the network. When a TTN is provided as input, TTNOpt reconstructs the network based on the EE without referencing the fidelity of the original state. We present three demonstrations of TTNOpt: (1) Ground-state search for the hierarchical chain model with a system size of 256. The entanglement patterns of the ground state manifest themselves in a tree structure, and TTNOpt successfully identifies the tree. (2) Factorization of a quantic tensor of the 2^{24} dimensions representing a three-variable function where each variant has a weak bit-wise correlation. The optimized TTN shows that its structure isolates the variables from each other. (3) Reconstruction of the matrix product network representing a 16-variable normal distribution characterized by a tree-like correlation structure. TTNOpt can reveal hidden correlation structures of the covariance matrix.

1. Introduction

Tensors, as multidimensional arrays, are widely used across various computational sciences, including condensed matter physics, big data analytics, and machine learning. A fundamental difficulty with manipulating tensors is that the number of tensor elements grows exponentially with the tensor rank N. One promising approach to overcoming this challenge is to employ the tensor network (TN) representation (decomposition), in which the tensor of interest is expressed as a contraction of small, low-rank tensors [3]. By setting an upper bound χ on the dimensions of each index (mode), which is referred to as auxiliary bonds, in $O(N)$ small tensors during factorization, the total number of elements in the TN can be reduced to $O(N\chi^p)$ where p reflects the maximum rank of small tensors. TNs have found broad applications in condensed matter physics and data science. In the former, high-rank tensors, such as wave functions and Boltzmann weights, are handled within the TN framework [4,5]. In the latter, TNs are utilized for representing complex data, including images [6,7]. Further expanding their applications are particularly in machine learning [8–10]. Quantic tensors have been used for enabling the treatment of functions with continuous variables [11,12]. So far, several types of TN structures have been developed, particularly in the quantum many-body physics [13–16].

PROGRAM SUMMARY
Program Title: TTNOpt
CPC Library link to program files: (to be added by Technical Editor)
Developer's repository link: Reference [1]
Licensing provisions: Apache 2.0
Programming language: Python
External routines/libraries: Reference [2]
Nature of problem:
Characterizing the entanglement structure of the lowest energy state of quantum spin systems and of high-dimensional tensor data, for efficient representation.
Solution method:
Tensor network contractions combined with a variational algorithm based on the Lanczos method, with automatic structural optimization

* Corresponding author.
E-mail address: ryowatanabe06jp@gmail.com (R. Watanabe).

https://doi.org/10.1016/j.cpc.2025.109918
Received 11 May 2025; Received in revised form 1 October 2025; Accepted 21 October 2025
Available online 29 October 2025

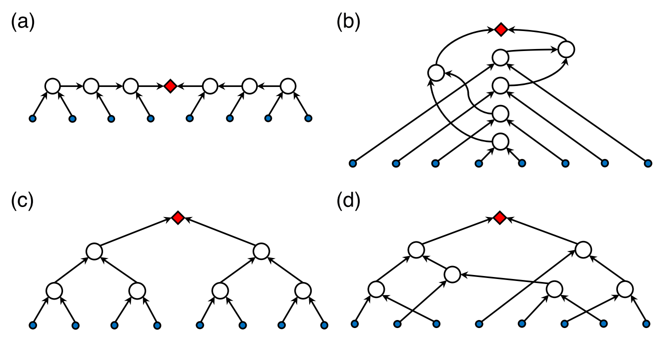

Fig. 1. Examples of TTN structures: (a) a matrix product network (MPN) or tensor train, (b) a rainbow structural network, (c) a perfect binary tree (PBT) network, and (d) a general TTN. White circles represent tensors defined in Eq. (10), while blue ones represent bare sites. Bare sites are arranged from left to right, following the order of basis states (indices). Arrows indicate tensor indices and point from the bare sites to the canonical center. These directions correspond to the domain and codomain of the isometric mapping as Eq. (10). The red square highlights the singular value tensor at the canonical center. Notably, the position of the singular value tensor can be arbitrary under gauge transformations [17,18]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

of tree tensor networks.
Restrictions:
Applicable to quantum spin systems and data that can be represented as tensors.
Unusual features:
Adaptive structural reconfiguration of TTNs based on the system's entanglement pattern.

In this study, we focus on the tree-tensor networks (TTNs) [19–21], which have no loop in their network structure; see Fig. 1 for the examples. The tree structure naturally allows us to impose the isometric conditions on tensors, resulting in efficient contraction schemes [14,22,23]. In particular, the isometric conditions guarantee that a TTN can be brought into the form of Schmidt decomposition across any bipartition regions A and B:

$$|\Psi\rangle = \sum_{p,q}\sum_{c} U_{pc} D_c V_{cq} |\psi_p\rangle^A |\psi_q\rangle^B \ , \tag{1}$$

where U and V are unitaries, and D is the singular values tensor. Here, TTN state $|\Psi\rangle$ belongs to the Hilbert space $\mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_B$, where $\{|\psi_p\rangle^A\}$ and $\{|\psi_q\rangle^B\}$ are orthonormal bases of the Hilbert spaces $\mathcal{H}_A$ and $\mathcal{H}_B$, respectively. This representation makes it possible to take the truncation of D_c. Additionally, the entanglement entropy (EE) of $|\Psi\rangle$ between A and B can be calculated as

$$\begin{aligned} S &= -\,\mathrm{Tr}\left[\rho_{\mathrm{A}} \log\left(\rho_{\mathrm{A}}\right)\right] = -\,\mathrm{Tr}\left[\rho_{\mathrm{B}} \log\left(\rho_{\mathrm{B}}\right)\right] \\ &= -\sum_c (D_c)^2 \log(D_c)^2 \ , \end{aligned} \tag{2}$$

where $\rho_A = \mathrm{Tr}_B(|\Psi\rangle\langle\Psi|)$ and $\rho_B = \mathrm{Tr}_A(|\Psi\rangle\langle\Psi|)$.

While introducing truncation of D_c by bounding the dimension of tensors with $c \in [1,\chi]$ on Eq. (1) reduces computational complexity, it also leads to a loss of precision in the TTN representation. Therefore, mitigating the loss of precision due to the finite bond dimension χ is a critical issue in the TN approach.

A promising solution to the problem is to optimize the network structure. The following examples illustrate the relevance of the network structure to the accuracy of the TTN approach. Let us consider a quantum state with a one-dimensional (1D) entanglement pattern, where the entanglement between qubits arranged in 1D is short-ranged [Fig. 2(a)]. For this state, a Matrix Product Network (MPN) [24,25], depicted in Fig. 1(a), also known as a tensor train [26], is a reasonable choice. As a result, MPN-based approaches, such as the density-matrix renormalization group (DMRG) method [27–29], works well

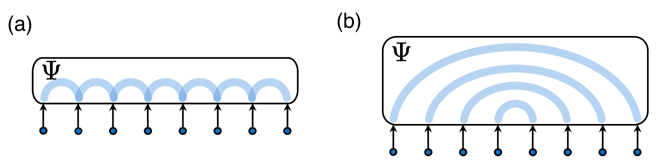

Fig. 2. Examples of the entanglement structure of wavefunctions Ψ: (a) a state with a one-dimensional entanglement structure, and (b) a state with a rainbow bell pairs structure.

for 1D quantum systems. As another example, consider a state in which Bell-paired qubits are arranged in a rainbow pattern [30,31], such as $\bigotimes_{j=1}^{N/2} \frac{1}{\sqrt{2}}\left(|0\rangle_j|1\rangle_{N+1-j} + |1\rangle_j|0\rangle_{N+1-j}\right)$ for even N, as shown in Fig. 2(b). If the MPN is applied to represent this state accurately, the canonical center in Fig. 1(a) must accommodate an exponentially large dimension χ with respect to N, in order to carry an amount of entanglement equivalent to $N/2$ Bell pairs. The bonds with a constant dimension χ miss such a large amount of entanglement, resulting in a significant loss of accuracy. This problem can be resolved by employing a TTN with the structure shown in Fig. 1(b). Using the TTN with this appropriate structure reduces the entanglement carried by each bond to that of, at most, a single Bell pair, allowing the state to be represented accurately with $\chi = 2$.

As demonstrated by the above examples, the network structure can significantly affect the efficiency of the TTN approach [32–36]. Nevertheless, identifying the optimal network structure remains a nontrivial problem, particularly for states with complex entanglement distributions. Several studies have been conducted to develop methods for determining optimal TTN structures. Many of these works have focused on optimizing the ordering of qudits in MPNs [37–41], while others have attempted to explore optimal structures within TTNs [42,43].

An algorithm for searching the optimal TTN structure in variational calculations of quantum many-body systems has been proposed [44]. This algorithm focuses on a particular bond in the TTN and recombines the local network structure to minimize the entanglement brought by that bond. By iterating this procedure while sweeping the entire network, the algorithm explores the TTN with the optimized structure. This algorithm has been proven effective for several quantum spin models [45,46], and data science [47].

In this work, we provide a software library for TTN calculations that includes the optimization of network structures. The library consists of three main packages:

(1) To perform the variational calculation for the lowest energy state of quantum spin systems. TTNOpt provides the variational wave function with the optimized structure in the TTN format. It outputs the optimized TTN, the variational energy, EE, and truncation error for each bond in the TTN, as well as the expectation values of various one- and two-spin functions. TTNOpt can treat a wide range of quantum spin models. Namely, the Hamiltonian terms that can be handled include the XXZ/XYZ exchange, Dzyaloshinskii-Moriya, and symmetric off-diagonal exchange interactions for arbitrary spin pairs, as well as external magnetic fields and single-ion anisotropy for arbitrary spins. The interaction parameters can take different values depending on pairs or spins. The spin sizes can also be site-dependent. If the model treated has the U(1) symmetry of conservation of total magnetization, the user can specify the total magnetization of the subspace in which the variational calculation is performed.
(2) To factorize or decompose a given high-dimensional tensor into a TTN with the optimized structure. The user can input the tensor as a multidimensional function. TTNOpt first decomposes the input tensor into the MPN and then performs the TTN structural optimization while maximizing the fidelity with the input tensor.
(3) To reconstruct the network of a given TTN. The user can input a tensor represented in the TTN format. TTNOpt performs structural re-

connection of the network for the TTN from the original input as the initial TTN. While the TTN structure is optimized through sweeps, the tensors are only updated by the singular value decomposition (SVD).

Additionally, TTNOpt contains descriptions of sample calculations to demonstrate its functionality and practical applications.

The structure of this paper is as follows. In Section 2, we give an overview of the methods provided by TTNOpt and their basic usage. We also describe the variables used for the calculation in detail. In Section 3, we explain the implemented algorithms in TTNOpt. While the explanation focuses on the structural optimization procedure proposed in Ref. [44], we also discuss the numerical techniques for TTN manipulations. Then, we showcase example benchmarks for each implemented method in Section 4. Finally, Section 5 concludes with an outlook on TTNOpt and explores prospective avenues for future development.

2. Basic usage of TTNOpt

Here, we provide a detailed explanation of how to use TTNOpt. This section describes the input files required mainly for ground-state searches and high-rank tensor factorizations. The sample input files for the demonstrations discussed in Section 4, including those used for the network reconstruction, are available in the “sample” directory of the GitHub repository [1].

2.1. Ground state search

The TTNOpt package has been developed for finite-size spin systems, including XXZ and XYZ Hamiltonians,

$$H_{\mathrm{XXZ}} = \sum_{i,j(>i)} J_{ij}\left(s_i^x s_j^x + s_i^y s_j^y + \Delta_{ij}^z s_i^z s_j^z\right) , \tag{3}$$

$$H_{\mathrm{XYZ}} = \sum_{i,j(>i)} \left(J_{ij}^x s_i^x s_j^x + J_{ij}^y s_i^y s_j^y + J_{ij}^z s_i^z s_j^z\right) , \tag{4}$$

where $\boldsymbol{s}_i = (s_i^x, s_i^y, s_i^z)$ is a spin operator at i th site on an N-site system, and $\{J_{ij}, \Delta_{ij}^z\}$ or $\{J_{ij}^x, J_{ij}^y, J_{ij}^z\}$ are the coupling parameters for the exchange interactions between the i th and j th spins. The size of spin $\boldsymbol{s}_i$ can be arbitrary for each site. In addition to the Hamiltonian Eqs. (3) and (4), TTNOpt can treat the interactions including magnetic field

$$H_{\mathrm{h}_\alpha} = \sum_i -\mathrm{h}_i^\alpha s_i^\alpha , \tag{5}$$

single-ion anisotropy

$$H_{\mathrm{D}} = \sum_i \mathrm{D}_i (s_i^z)^2 , \tag{6}$$

Dzyaloshinskii-Moriya (DM) interaction

$$H_{\mathrm{DM}_\alpha} = \sum_{i,j(>i)} \mathrm{D}_{ij}^\alpha (\boldsymbol{s}_i \times \boldsymbol{s}_j)^\alpha , \tag{7}$$

and symmetric off-diagonal exchange anisotropy

$$H_{\Gamma_\alpha} = \sum_{i,j(>i)} \Gamma_{ij}^\alpha (s_i^\zeta s_j^\eta + s_i^\eta s_j^\zeta) , \tag{8}$$

where $\alpha, \zeta, \eta \in \{x, y, z\}$ and $\alpha \neq \zeta \neq \eta$.

Regarding the whole Hamiltonian $H = H_{\mathrm{XXZ}} + \sum_{\alpha\in\{x,y,z\}} H_{\mathrm{h}_\alpha} + H_{\mathrm{D}} + \sum_{\alpha\in\{x,y,z\}} H_{\mathrm{DM}_\alpha} + \sum_{\alpha\in\{x,y,z\}} H_{\Gamma_\alpha}$, if and only if it meets $\mathrm{h}_i^x = \mathrm{h}_i^y = \mathrm{D}_{ij}^x = \mathrm{D}_{ij}^y = \Gamma_{ij}^x = \Gamma_{ij}^y = \Gamma_{ij}^z = 0$, the Hamiltonian commutes with the operator for the total magnetization of z axis:

$$M = \sum_i s_i^z , \tag{9}$$

i.e., $[H, M] = 0$, so that the U(1) symmetry is preserved. In this case, TTNOpt provides a function to calculate the TTNs for the lowest-energy state within the subspace labeled by M.

2.1.1. How to set input files

Running the TTNOpt package requires the main input file and, if necessary, several setting files. The main input file consists of **system**, **numerics**, and **output**. The meaning of each section and the variables used there are explained below.

system

This section requires users to specify the information of the Hamiltonian for the target system, including the number of spins, spin size, and interactions in the Hamiltonian. TTNOpt requires users to prepare a separate file to define the two-site interactions of Eq. (3) or (4). Other additional terms Eqs. (5)–(8) are not necessarily specified if they are not present in the Hamiltonian. TTNOpt assumes that the input files are in the “.dat” format.

- **N** (INTEGER):
  The number of spins, N. Thus, TTN wave functions are tensors in the vector product space of N local spin Hilbert spaces.

- **spin_size** (REAL or STRING)
  The spin sizes s_i defined for $i \in [0, N-1]$. If a spin value s is provided, it is applied uniformly across the entire system, i.e., $s_i = s$ for all $i \in [0, N-1]$. Alternatively, if a file path is provided, TTNOpt imports the file, which must contain two columns: The first column specifies the site index i, and the second column specifies the corresponding spin value s_i. Note that, in the case that the spin size is a half-odd integer, TTNOpt does not accept decimal values for spin settings, and only the fractional representations, such as $1/2$ and $3/2$, are allowed.

- **model.type** (XXZ or XYZ)
  The basic interaction type, XXZ or XYZ interaction, that is respectively given in Eq. (3) or (4).
- **model.file** (STRING)
  The coupling parameters for the exchange interactions. Each row contains two integers and two or three floats, where the first two columns specify a pair of site indices i, j and subsequent columns specify the coupling parameters J_{ij}, Δ_{ij}^z or $J_{ij}^x, J_{ij}^y, J_{ij}^z$ according to **system.model.type**.
- **MF_X (Y, Z)** (REAL or STRING)
  The magnetic field along the α direction $(\alpha = x, y, z)$ described in Eq. (5). If a float value h is provided, it is applied uniformly across the entire system, i.e., $\mathrm{h}_i^\alpha = \mathrm{h}$ for $i \in [0, N-1]$. Alternatively, if the path to a file is provided, TTNOpt imports the file, which must contain two columns where the first and second columns specify, respectively, the site i and the corresponding value h_i^α.
- **SIA** (REAL or STRING)
  The single-ion anisotropy described in Eq. (6). If a float value D is provided, the anisotropy is applied uniformly across the entire system, i.e., $\mathrm{D}_i = \mathrm{D}$ for $i \in [0, N-1]$. Alternatively, if the path to a file is provided, TTNOpt imports the file, which must contain two columns where the first and second columns specify, respectively, the site i and the corresponding value D_i.
- **DM_X (Y, Z)** (STRING)
  The Dzyaloshinskii-Moriya (DM) interaction described in Eq. (7). TTNOpt requires a file with three columns to contain two integers and a real value. The first two columns identify a pair of site indices i, j, and the last column specifies D_{ij}^α.
- **SOD_X (Y, Z)**(STRING)
  The symmetric off-diagonal anisotropic exchange interaction described in Eq. (8). TTNOpt requires a file with three columns to contain two integers and a real value. The first two columns identify a pair of site indices i, j, and the last column specifies Γ_{ij}^α.

numerics

This section requires users to specify the conditions and hyperparameters for the calculation, including the settings for the structural optimization algorithm and the maximum bond dimension of the TTN states.

- **init_tree** (0 or 1)
If this value is set to 0, the initial structure is set to the MPN. If it is set to 1 and the system size is a power of 2, the initial structure is set to the perfect binary tree (PBT) structure. Otherwise, the MPN structure is used by default.
- **initial_bond_dimension** (INTEGER)
The maximum bond dimension χ_{init} during the preparation of an initial TTN. The initial tensors are prepared by the real space renormalization group (RSRG) [48,49], where the bond dimension of tensors is upper-bounded by χ_{init} (see Section 3.2.2 for details). The value of χ_{init} must not be too small to ensure that each tensor contains all degenerate lowest-energy states of the block Hamiltonian [Eq. (13) in Section 3.2.1] in the renormalized region belonging to the tensor.
- **total_magnetization** (REAL)
The total magnetization M that defines the subspace in which the lowest-energy state is computed. This input is used only for the Hamiltonian with U(1) symmetry. To activate this function, it is not allowed to use XYZ interaction in **system.model.type** even if the user numerically sets $J_x = J_y$. Among other terms, those that break the symmetry are also prohibited.
When the initial bond dimension χ_{init} is not sufficiently large, the RSRG method may fail to construct the initial TTN in the subspace labeled by the magnetization M. In this case, TTNOpt initializes the TTN state with a magnetization M' such that $|M'| < |M|$ and $|M'|$ is the closest to $|M|$ among the magnetizations that can be spanned by the RSRG. It then performs warmup calculations while keeping the TTN state's bond dimension at χ_{init} and adjusting M' as $M' := M' \pm 1$ at the end of each update sweep [50], until the appropriate initial state with magnetization M is realized.
- **opt_structure.type** (0, 1 or 2)
The method for optimizing the network structure. If the value is 0, TTNOpt does not optimize the TTN structure. If it is 1, the TTNOpt performs structural reconstruction by referring to EEs. If the value is 2, TTNOpt selects the structure with the minimum truncation error [Eq. (24) in Section 3.2.3]. However, when the differences between the minimum truncation errors are less than 1×10^{-13}, the optimal structure is determined using EE as a secondary criterion.
- **opt_structure.temperature** (REAL)
An effective temperature T_0, which is a positive real number. When **opt_structure.type** $= 1$, the structure is selected stochastically using this T_0 with n_τ, which is described later in Section 3.2.3. This value is set to $T_0 = 0$ by default, and then TTNOpt chooses the structure with the minimum EE.
- **opt_structure.tau** (INTEGER)
The decay factor n_τ, which is a positive integer. When **opt_structure.type** $= 1$ and $T_0 > 0$, it controls the temperature decay according to the sweep count n. [See Eq. (23) in Section 3.2.3.] TTNOpt sets n_τ to $\lfloor n_{\max}, 0/2 \rfloor$ by default.
- **opt_structure.seed** (INTEGER)
The random seed for stochastic selection of the structure. [See Eq. (22) in Section 3.2.3.] TTNOpt sets this value to 0 by default to ensure that the results are reproducible.
- **max_bond_dimensions** (LIST of INTEGER)
The elements of the list specify the maximum bond dimensions $\chi = [\chi_m]_{1 \le m \le \mathfrak{m}}$ in the TTN, where $\mathfrak{m}$ is the total number of stages. At the mth stage, TTNOpt performs sweeps using $\chi := \chi_m$ until the TTN state converges or the number of sweeps reaches $n_{\max,m}$, as specified in **numerics.max_num_sweeps**. Once the sweeps at stage m are completed, TTNOpt proceeds to the next stage with $\chi := \chi_{m+1}$ and $n_{\max,m+1}$, using the TTN state obtained at $\chi = \chi_m$ as the initial state. It is worth noting that the values in χ should be arranged in ascending order. This strategy enables computation with larger χ_m, which requires a higher computational cost, to begin from a well-prepared initial state. As a result, the number of sweeps needed for convergence at larger χ_m can be reduced. If **numerics.opt_structure.type** $\in \{1, 2\}$, the structural optimization is applied *only for* the sweeps with $\chi = \chi_1$. As for the remaining computation with $\chi = \chi_m$ for $m > 1$, only tensors are updated while the structure is fixed.
- **max_num_sweeps** (LIST of INTEGER)
The elements of the list specify the maximum number of sweeps $\boldsymbol{n}_{\max} = [n_{\max,m}]_{1 \le m \le \mathfrak{m}}$ for each stage of calculations. All elements $n_{\max,m}$ should be set sufficiently large to achieve the TTN state's convergence. In this paper, we denote $n_{\max,m}$ simply as $n_{\max}$ to represent the maximum number of sweeps at m th stage unless otherwise specified.
- **energy_convergence_threshold** (REAL)
The tolerance ϵ_E for the convergence of energy. If the relative difference between each the energy $\mathfrak{E}_b$ calculated by using the Lanczos method for the auxiliary bond b at the current sweep and that from the previous sweep, $\mathfrak{E}'_b$, is less than the threshold ϵ_{E}, i.e., $|1 - \frac{\mathfrak{E}'_b}{\mathfrak{E}_b}| < \epsilon_{\mathrm{E}}$ for all b, TTNOpt considers that the TTN has been converged concerning energy. TTNOpt sets this value to $\epsilon_{\mathrm{E}} = 1 \times 10^{-8}$ by default.
- **entanglement_convergence_threshold** (REAL)
The tolerance ϵ_S for the convergence of EE. If the difference between the bipartite EE $\mathfrak{S}_b$ for the bond b at the current sweep as Eq. (2) and that from the previous sweep, $\mathfrak{S}'_b$, is less than the threshold ϵ_S, i.e., $|\mathfrak{S}_b - \mathfrak{S}'_b| < \epsilon_S$ for all bonds b, TTNOpt considers that the EEs of TTN have been converged. Furthermore, if **opt_structure.type** is set to 1 and the EE of the optimal and previous structures differs by less than ϵ_S, the structure remains unchanged to avoid inconsequential fluctuation of the TTN structure at each step of sweeps. TTNOpt sets this value to $\epsilon_S = 1 \times 10^{-8}$ by default.
- **energy_degeneracy_threshold** (REAL)
The threshold δ_{E} used in the preparation of the initial tensors by the RSRG to determine whether the eigenvalues of a block Hamiltonian are degenerate. The procedure is detailed in Section 3.2.2. TTNOpt sets this value to $\delta_{\mathrm{E}} = 1 \times 10^{-8}$ by default.
- **entanglement_degeneracy_threshold** (REAL)
The threshold δ_S used in the SVD to determine whether singular values are degenerate for updating the local two-tensor. The procedure is detailed in the last paragraph of Section 3.2.2. TTNOpt sets this value to $\delta_S = 1 \times 10^{-8}$ by default.

output
This section requires users to specify the physical quantities for which TTNOpt will generate output files. By default, TTNOpt outputs, in a file named "basic.csv", the EEs for all bonds, as well as the variational energies and truncation errors for the auxiliary bonds, computed during the final sweep of each stage $m \in [1, \mathfrak{m}]$. In the "basic.csv" file, all bonds are identified by two nodes connected by the bond, (i, i') with $i, i' \in [0, N + N_{\mathrm{t}} - 1]$, where N is the number of spins and N_{t} is the number of tensors. Additionally, TTNOpt saves the set of bond labels $\boldsymbol{E}$ for the TTN structure in a file named "graph.dat". The format to describe the bond labels is explained in the second paragraph of Section 3.1. Users can save single-site spin expectation values at each site and two-site spin correlations between any two sites.

- **dir** (STRING)
The location of the directory where the data will be output.
- **single_site** (0 or 1)
If this value equals 1, TTNOpt calculates single-site spin expectation values and saves them in a file named "single_site.csv". The file has four columns where the first column owns site i with $i \in [0, N-1]$ and the subsequent columns have $\langle s_i^\alpha \rangle$ with $\alpha = x, y, z$, where $\langle \cdots \rangle$ denotes the expectation value of $\cdots$ in the ground state. Otherwise, TTNOpt does not save them.
- **two_site** (0 or 1)
If this value equals 1, TTNOpt calculates two-site spin correlation

functions and saves them in a file named “two_site.csv”. The file has eleven columns where the first two columns own a pair of two sites i, j with $i, j \in [0, N-1]$ and $i < j$, and the subsequent columns have $\langle s_i^\alpha s_j^\beta \rangle$, where $(\alpha, \beta) \in \{xx, yy, zz, yz, zy, zx, xz, xy, yx\}$. Otherwise, TTNOpt does not save them.

2.1.2. Run and results

After preparing all input files described above, users can perform the calculation as follows:

```
$ gss input.yml
```

Here, the results of m th stage of calculation are saved in a subdirectory “run$\{m\}$”, where $\{m\}$ with $m \in [1, \mathfrak{m}]$ relies on variable expansion of Python notation, under the directory specified by the **output.dir** in the main input file.

2.2. Factorising tensors

The TTNOpt package provides functions to factorize a high-dimensional tensor $\Psi = \{\Psi_{s_0,\ldots,s_{N-1}}\}$, with indices $s_0, \ldots, s_{N-1}$, into a TTN as an efficient data structure. The overall procedure is illustrated in Fig. 3.

TTNOpt first decomposes the input tensor Ψ into an MPN using sequential SVD [Fig. 3(c)] [29] . Users may then choose to transform this MPN into a TTN (to perform structural optimization) via reconstruction sweeps [Fig. 3(d)]; this transformation performs local reconstruction of a tree structure, thereby reducing the bipartite EEs across the TTN internal bonds. We detail these procedures in Section 3.3.

If the entanglement structure of the input tensor Ψ is not compatible with the MPN, the TTN obtained from the procedures above may suffer from low-precision approximations. To address this limitation, TTNOpt also implements the fidelity-based optimization that directly refers to Ψ [Fig. 3(e)]. Specifically, the fidelity-based optimization updates the tensor elements to maximize the fidelity with Ψ, while simultaneously performing the structural optimization. However, since it requires explicit contractions with Ψ at each update step, it entails a significantly higher computational cost compared to the reconstruction process [Fig. 3(d)] alone.

We note that the index order of the input tensor plays a crucial role in the accuracy of converting Ψ into an MPN [Fig. 3(c)] and subsequently into a TTN [Fig. 3(d)]. In principle, the fidelity-based optimization [Fig. 3(e)] can cope with this issue. However, the index order might affect the performance of this optimization. In practical calculations, the optimization may be trapped in a local minimum or encounter slow convergence. To mitigate these issues, TTNOpt provides several structural optimization options. In particular, a probabilistic selection strategy with an effective temperature is effective for avoiding local traps [46,51].

2.2.1. How to set input files

target

This section requires specifying the directory of the input tensor Ψ. TTNOpt needs users to specify “.npy” format file as tensors.

- **tensor** (STRING)
  The directory name for the file of the input tensor Ψ.

numerics

This section requires users to specify the conditions for the calculation.

- **initial_bond_dimension** (INTEGER)
  The initial bond dimension χ_{init}. TTNOpt first decomposes Ψ into an MPN structure with up to this bond dimension χ_{init} using the SVD.
- **opt_structure.type** (0, 1 or 2)
  This value is applied to local reconstructions in TTN structural optimization. Explantions of this input and the following ones, **opt_structure.temperature**, **opt_structure.tau**, and **opt_structure.seed** are in Section 2.1.1.
- **max_sweep_num** (INTEGER)
  The maximum number of sweeps n_{max} for reconstruction. During the sweep procedure, we set the bond dimension χ as χ_{init} specified by **initial_bond_dimension**.
- **entanglement_convergence_threshold** (REAL)
  The entanglement convergence threshold ϵ_S, which is explained in Section 2.1.1. It is noted that this value is used both when TTNOpt reconstructs the initial MPN into the optimal TTN and updates the TTN state based on the fidelity with the input tensor Ψ.
- **max_truncated_singularvalue** (REAL)
  This threshold σ is used in SVD to reduce the bond dimension while tolerating a certain loss of accuracy. All singular values satisfying $D_i/D_1 \le \sigma$ are truncated, where D_i is the ith singular value sorted in descending order. By default, σ is set to 0, in which case up to χ singular values are retained.
- **fidelity.opt_structure.type** (0, 1 or 2)
  This value is applied when TTNOpt updates the TTN state according to the fidelity. Explantions of this value and the following inputs, **fidelity.opt_structure.temperature**, **fidelity.opt_structure.tau**, **fidelity.opt_structure.seed**, and **fidelity.max_num_sweeps** are in Section 2.1.1. Note that if users set values related to the fidelity-based optimization in the input file, TTNOpt will update both the network structure and tensors of the TTN concerning Ψ, even if the TTN structure has already been determined by **opt_structure.type**.
- **fidelity.max_bond_dimensions** (LIST of INTEGER)
  Explanations of this value are detailed in Section 2.1.1. The bond dimension practically kept can be reduced when a nonzero σ is set in **max_trunated_value**.
- **fidelity.convergence_threshold**(REAL)
  The tolerance ϵ_F for the convergence of fidelity. If the difference between the fidelity F_b calculated for auxiliary bond b at the current sweep and that from the previous sweep, F_b', is less than the threshold ϵ_F, i.e., $|F_b - F_b'| < \epsilon_F$ for all auxiliary bonds b, TTNOpt considers that the TTN has been converged concerning fidelity.

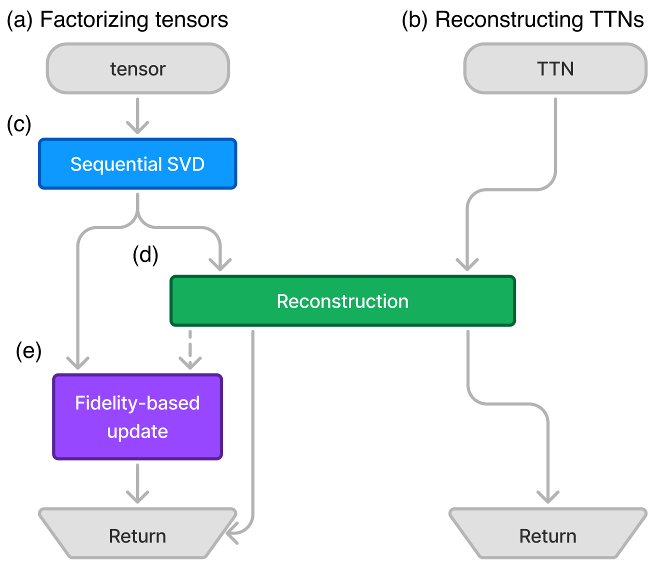

Fig. 3. Schematic overview of the procedures for (a) factorizing tensors and (b) reconstructing TTNs. In (a), TTNOpt performs (c) sequential SVD to construct an MPN, after which users might either (d) apply reconstruction to this MPN or proceed directly to (e), where both the tensors and the network structure of the TTN are optimized to maximize the fidelity with Ψ. TTNOpt also allows using the optimized TTN obtained in (d) as the input to (e), depicted as a dashed line. In (b), TTNOpt runs (d) for the given TTN.

output

This section requires users to specify the output settings. By default, TTNOpt outputs properties of TTN in a file named “basic.csv” and TTN structure in “graph.dat” in the same manner as the ground state search algorithm. In the “basic.csv”, TTNOpt outputs the EEs for all bonds and the truncation errors for all auxiliary bonds. TTNOpt also saves the fidelity with the input tensor Ψ calculated at each auxiliary bond to a file named “basic.csv”, if users specify the input file to conduct the update regarding the fidelity.

In addition to the outputs above, the user can save the tensors, singular values, and the norm of the tensor Ψ, which is used for the normalization of the TTN.

- **dir** (STRING)
  The location of the directory where the data will be output.
- **tensors** (0 or 1)
  If this value is 1, TTNOpt saves optimized tensors as the “isometry{i}.npy” file with $i \in [0, N_t - 1]$, singular values tensor in the “singular_values.npy” file, and norm in the “norm.npy” file. Otherwise, TTNOpt does not save them.

2.2.2. Run and result

After preparing the input files described above, users can perform the calculation as follows:

```
$ ft input.yml
```

Here, the computed results are output in the directory specified by the **output.dir** variable in the main input file.

2.3. Reconstruction of TTNs

Motivated by the recent applications of MPNs and tensor train data, TTNOpt allows users to load a TTN. TTNOpt then reconstructs the network structure of a given TTN [Fig. 3(b) and (d)]. This function is executed by reconstruction sweeps introduced in Section 2.2; for the details, the user can refer to Section 3.3.

This function of TTNopt would be powerful for searching more efficient TTN structures in the sense that each bond carries low EE or has small bond dimensions.

Users can perform the calculation as follows:

```
$ ft input.yml
```

Note that the command to execute this method is identical to the one used for the factorizing tensors. The users must ensure that the input tensor is consistent with the intended type, which is either a tensor or a TTN (see the sample input files in Ref. [1]). The variables for the calculations share the same format as those for the factorizing tensors method, excluding those relevant to the fidelity-based optimization, described in Section 2.2.1.

3. Inplemented algorithms

3.1. Representation of TTNs

The TTNOpt package constructs TTN states from a set of three-leg isometric tensors $\boldsymbol{v} = \{v_0, v_1, \ldots, v_{N_t-1}\}$, where $N_t = N - 2$ is the total number of tensors [52]. Each isometric tensor $[v_i]^{i_3}_{i_1 i_2}$ for all $0 \le i \le N_t - 1$, where i_k with $k \in \{1, 2, 3\}$ represents index with bond dimensions χ_{i_k}, satisfies the following isometric condition:

$$\sum_{i_1 i_2} [v_i]^{i_3}_{i_1 i_2} [v_i^*]^{i'_3}_{i_1 i_2} = \delta_{i_3 i'_3}\,, \tag{10}$$

where v_i^* denotes the complex conjugate of v_i, and $\delta_{i_3 i'_3}$ is the Kronecker delta. On the isometry $[v_i]^{i_3}_{i_1 i_2}$, the degrees of freedom χ_{i_3} is limited by the product of the bond dimensions χ_{i_1} and χ_{i_2}, i.e., $\chi_{i_3} \le \chi_{i_1}\chi_{i_2}$. In addition, TTNOpt practically poses an upper bound χ on bond dimensions for all bonds in the TTN state.

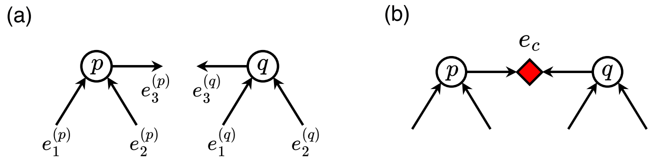

Fig. 4. Schematic diagrams of (a) data structure of labels for tensors and edges around the initial canonical center, i.e., $e_3^{(p)} = e_3^{(q)}$, and (b) local structure surrounding the canonical center e_c.

TTNOpt maintains connectivity between tensors as a set of three bond (edge) labels $\boldsymbol{E} = \{E_0, \ldots, E_{N_t-1}\}$ where $E_i = (e_1^{(i)}, e_2^{(i)}, e_3^{(i)})$, to define TTN structures. Each edge label e is written as an integer $e \in [0, 2N_t]$ where $2N_t + 1$ is the number of bonds in the TTN. The edge with the label $e = r \in [0, N - 1]$ is connected to the bare spin s_r. In this data structure, two tensors v_i and v_j that share the same edge label in E_i and E_j are connected. There is an unique pair of tensors specified by p and q connected to each other through $e_3^{(p)}$ and $e_3^{(q)}$ as shown in Fig. 4(a). This bond is referred to as the canonical center e_c, defined as $e_c = e_3^{(p)} = e_3^{(q)}$ detailed in Fig. 4(a) and (b). It is important to note that in TTNOpt, e_c is supposed not to be connected to any physical sites, i.e., $e_c \notin [0, N - 1]$. TTNOpt assumes that up to χ elements from the vector $D = (D_1, \ldots, D_{\chi^2})$ are assigned on e_c, where the singular values in D are ordered in descending magnitude as $D_1 \ge D_2 \ge \cdots \ge D_{\chi^2} \ge 0$. Since the truncation of bond dimensions causes a loss of the norm of TTN states, TTNOpt rescales the singular values, $\mathfrak{D}_c$ with $c \in [1, \chi]$ as

$$\mathfrak{D}_c := \frac{D_c}{\sqrt{\sum_{c'=1}^{\chi} (D_{c'})^2}}. \tag{11}$$

That is because, within the canonical formulas, the norm of the TTN state $|\Psi\rangle$ is described as $\langle\Psi|\Psi\rangle = \sum_{c'=1}^{\chi} (D_{c'})^2$. With these settings, TTNOpt allows for representing TTN states in mixed canonical form [29] to manage various calculations efficiently.

3.2. Ground state search

3.2.1. Representation of the Hamiltonian

Conducting the ground-state search requires constructing the effective Hamiltonian for each renormalized region, which is achieved by contracting a tensor network composed of the TTN state and the local Hamiltonian tensors. For a general TTN structure, this procedure differs from the case of an MPN, where the full contraction with matrix product operators, whose bond dimension is $O(1)$, can be carried out efficiently with a computational cost of $O(N\chi^3)$ [29]. TTNOpt preserves the renormalized spin operators, $\tilde{S}^z$ and $\tilde{S}^+$, at all bonds, enabling efficient evaluation of these contractions. Since $\tilde{S}^-$ is the Hermitian conjugate of $\tilde{S}^+$, it is sufficient to retain only the $\tilde{S}^z$ and $\tilde{S}^+$ operators [27,28]. In following paragraphs, we define the renormalized spin operators on a TTN and describe their construction, and then show how we construct the effective Hamiltonians.

To manage the calculation on TTN, we assign a distance $d_{e_3^{(i)}} \in [0, N_t - 1]$ to each edge label $e_3^{(i)}$ for all v_i, measured from the origin bond o_c, which corresponds to the canonical center of the initialized TTN and $d_{o_c} = 0$. Specifically, the value $d_{e_3^{(i)}}$ represents the minimum number of isometries that must be traversed to reach v_i from o_c. We also introduce $\boldsymbol{L} = \{\boldsymbol{l}_0, \boldsymbol{l}_1, \ldots, \boldsymbol{l}_{d_{\max}}\}$ with $\boldsymbol{l}_{\mathfrak{d}} = \{l_{\mathfrak{d}}^1, \ldots, l_{\mathfrak{d}}^{n_{\mathfrak{d}}} \mid l_{\mathfrak{d}}^i \in [0, N_t - 1]\}$, where $n_{\mathfrak{d}}$ is the number of edges whose distances are equal to $\mathfrak{d}$, i.e., $d_{l_{\mathfrak{d}}^1} = d_{l_{\mathfrak{d}}^2} = \cdots = d_{l_{\mathfrak{d}}^{n_{\mathfrak{d}}}} = \mathfrak{d}$). Regarding the maximum distance $d_{\max}$, the isometries $\{v_i \mid e_3^{(i)} \in \boldsymbol{l}_{d_{\max}}\}$ are ensured to connect directly to bare sites, i.e., $e_i^{(1)}, e_i^{(2)} \in [0, N - 1]$. For example, in the MPN structure as Fig. 1(a), in which the origin bond o_c is positioned on the midpoint of the system, $\{v_i \mid e_3^{(i)} \in \boldsymbol{l}_{d_{\max}}\}$ correspond to two isometries located at both ends of the MPN. Additionally, to define $\boldsymbol{L}$ uniquely, we incorporated an ordering rule such that $l_{\mathfrak{d}}^1 < l_{\mathfrak{d}}^2 < \cdots < l_{\mathfrak{d}}^{n_{\mathfrak{d}}}$ for each $\boldsymbol{l}_{\mathfrak{d}}$. We finally introduce a set of spin locations $\boldsymbol{r}_k^{(i)} = \{r_{k,1}^{(i)}, r_{k,2}^{(i)}, \ldots, r_{k,g(i,k)}^{(i)}\}$ within the renormalized region specified by the edge label $e_k^{(i)}$, where $g(i,k) = |\boldsymbol{r}_k^{(i)}|$. It is trivial that $\boldsymbol{r}_3^{(i)} = \boldsymbol{r}_1^{(i)} \cup \boldsymbol{r}_2^{(i)}$, since the sets of spins of $e_1^{(i)}$ and $e_2^{(i)}$ are renormalized to $e_3^{(i)}$ by v_i.

These definitions allow us to consider the following renormalized spin transformation using the isometry v_i:

$$\left[\tilde{S}^{(\cdot)}_{e_3^{(i)},r}\right]_{i_3,i_3'} = \begin{cases} \displaystyle\sum_{i_1 i_2 i_1'} \left[v_i\right]^{i_3}_{i_1,i_2} \left[\tilde{S}^{(\cdot)}_{e_1^{(i)},r}\right]_{i_1 i_1'} \left[v_i^*\right]^{i_3'}_{i_1',i_2} & (r \in \boldsymbol{r}_1^{(i)}) \\ \displaystyle\sum_{i_1 i_2 i_2'} \left[v_i\right]^{i_3}_{i_1,i_2} \left[\tilde{S}^{(\cdot)}_{e_2^{(i)},r}\right]_{i_2 i_2'} \left[v_i^*\right]^{i_3'}_{i_1,i_2'} & (r \in \boldsymbol{r}_2^{(i)}) \end{cases}, \tag{12}$$

where $\tilde{S}^{(\cdot)}_{e_1^{(i)},r}$ and $\tilde{S}^{(\cdot)}_{e_2^{(i)},r}$ are sandwiched by the isometry v_i and its Hermitian conjugate and projected onto the reduced subspace on $e_3^{(i)}$. If $e_k^{(i)} = r$, $\tilde{S}^{(\cdot)}$ is equal to the bare spin operator $s^{(\cdot)}$ where $(\cdot) \in \{z, +\}$ indicates the type of spin operators. We then calculate renormalized spin operators on $d_{e_3^{(i)}} \in d_{\max}$ by using Eq. (12). Applying the renormalization procedure of Eq. (12) recursively enables to compose further block-spin operators on the edges in $\boldsymbol{l}_{\mathfrak{d}-1}$ with $\{v_i \mid e_3^{(i)} \in \boldsymbol{l}_{\mathfrak{d}}\}$ from $\mathfrak{d} = d_{\max}$ to $\mathfrak{d} = 0$.

Using the block-spin operators, we can construct the block Hamiltonian associated with v_i of, for example, the XXZ model of Eq. (3):

$$\tilde{H}_{e_1^{(i)} e_2^{(i)}} = \tilde{H}_{e_1^{(i)}} + \tilde{H}_{e_2^{(i)}} + \tilde{H}^{\text{int}}_{e_1^{(i)} e_2^{(i)}},$$
$$\tilde{H}^{\text{int}}_{e_1^{(i)} e_2^{(i)}} = \sum_{r \in \boldsymbol{r}_1^{(i)}} \sum_{r' \in \boldsymbol{r}_2^{(i)}} \frac{J_{rr'}}{2} \left(\tilde{S}^{+}_{e_1^{(i)},r} \tilde{S}^{-}_{e_2^{(i)},r'} + \tilde{S}^{-}_{e_1^{(i)},r} \tilde{S}^{+}_{e_2^{(i)},r'} + \Delta^z_{rr'} \tilde{S}^{z}_{e_1^{(i)},r} \tilde{S}^{z}_{e_2^{(i)},r'} \right). \tag{13}$$

Then, we get

$$\left[\tilde{H}_{e_3^{(i)}}\right]^{i_3}_{i_3'} = \sum_{i_1 i_2 i_1' i_2'} \left[v_i\right]^{i_3}_{i_1 i_2} \left[\tilde{H}_{e_1^{(i)} e_2^{(i)}}\right]_{i_1 i_2 i_1' i_2'} \left[v_i^*\right]^{i_3'}_{i_1' i_2'}. \tag{14}$$

The expression in Eq. (14) describes the projection of $\tilde{H}_{e_1^{(i)} e_2^{(i)}}$ onto the reduced subspace on $e_3^{(i)}$. The extension to a general Hamiltonian including up to two-body interactions is straightforward.

Finally, the superblock-effective Hamiltonian corresponding to two adjacent tensors $\{v_p, v_q \mid \{p,q\} = \boldsymbol{l}_0\}$ surrounding the origin bond o_c can be described as:

$$\tilde{H}_{e_1^{(p)} e_2^{(p)} e_1^{(q)} e_2^{(q)}} = \sum_{k \in \{1,2\}} \sum_{r \in \{p,q\}} \tilde{H}_{e_k^{(r)}} + \sum_{r \in \{p,q\}} \tilde{H}^{\text{int}}_{e_1^{(r)} e_2^{(r)}} + \sum_{k,k' \in \{1,2\}} \tilde{H}^{\text{int}}_{e_k^{(p)} e_{k'}^{(q)}}. \tag{15}$$

3.2.2. Initializing TTN tensors

Given a TTN structure identified by $\boldsymbol{E}$, TTNOpt initializes isometric tensors $\boldsymbol{v}$ using the real-space renormalization group (RSRG) [48,49]. Namely, TTNOpt decides isometries following a recursive sequence from physical bonds to the canonical center of the initial TTN, o_c, in the same order of composing renormalized spin operators by referring to $\boldsymbol{L}$ as introduced in the previous section.

In order to initialize the isometry v_i, TTNOpt requires the full diagonalization of $\tilde{H}_{e_1^{(i)} e_2^{(i)}}$ in Eq. (13). The corresponding eigenvectors $\boldsymbol{u}$ are then collected in ascending order of their eigenvalues, and we obtain the element of isometry by

$$\left[v_i\right]^{i_3}_{i_1 i_2} := \left[\boldsymbol{u}\right]^{i_3}_{i_1 i_2}, \tag{16}$$

where $\left[\boldsymbol{u}\right]^{i_3}_{i_1 i_2}$ is the i_3th eigenvector reshaped into a two-dimensional tensor indexed by i_1 and i_2. The maximum bond dimension here is χ_{init}, which is determined by **numerics.initial_bond_dimension**. Consequently, the computational cost of full diagonalization of the block Hamiltonian is up to $O(\chi_{\text{init}}^6)$.

It is worth noting that TTNOpt selects up to χ_{init} eigenvectors, accounting for degeneracies arising from symmetries of the system [27,28]. In the case of degenerate eigenvectors, they are either fully retained or discarded to maintain symmetry. To detect degeneracies, TTNOpt calculates the L1 norm of energy differences

$$\Delta_k = \left| \mathrm{e}^{k+1}_{e_3^{(i)}} - \mathrm{e}^{k}_{e_3^{(i)}} \right|, \tag{17}$$

where $\mathrm{e}^{k}_{e_3^{(i)}}$ is the kth eigenvalue of the block Hamiltonian $\tilde{H}_{e_1^{(i)} e_2^{(i)}}$ for k ranging from $\chi_{\text{init}} - 1$ to 0. If $\Delta_k < \delta_{\text{E}}$, where δ_{E} is set by **numerics.energy_degeneracy_threshold**, TTNOpt discards the $(k+1)$th eigenvector and updates $k := k - 1$. The iteration continues until $\Delta_k \geq \delta_{\text{E}}$ and then the practical bond dimension of i_3 is determined as $\chi_{e_3^{(i)}} = k + 1 \in [1, \chi_{\text{init}}]$.

Once the RSRG flow is complete, in order to determine singular values tensor $\mathfrak{D}$, TTNOpt derives the renormalized wave function $\tilde{\Psi}_{e_1^{(p)} e_2^{(p)} e_1^{(q)} e_2^{(q)}}$ and bond energy $\mathfrak{E}_{o_c}$ from the diagonalization of the superblock Hamiltonian $\tilde{H}_{e_1^{(p)} e_2^{(p)} e_1^{(q)} e_2^{(q)}}$ of Eq. (15) by using the Lanczos method. TTNOpt then performs the SVD for $\tilde{\Psi}$ as follows:

$$\left[\tilde{\Psi}\right]_{p_1 p_2 q_1 q_2} = \sum_c U^c_{p_1 p_2} D_c V^c_{q_1 q_2}, \tag{18}$$

where c is up to χ_{init}^2. TTNOpt takes $\chi' \leq \chi_{\text{init}}$ singular values considering degeneracies of singular values in the same manner as energy degeneracies while, in this case, TTNOpt compares the relative variation $\Delta_k^{\text{rel}} = \frac{D_{k+1} - D_k}{D_k}$, with $k \in [0, \chi_{\text{init}} - 1]$, to δ_S defined by **numerics.entanglement_degeneracy_threshold**. After the bond dimension χ_{o_c} is decided, TTNOpt replaces two isometries $\{v_p, v_q \mid \{p,q\} = \boldsymbol{l}_0\}$ such that

$$\left[v_p\right]^{p_3}_{p_1 p_2} := \left[U\right]^{p_3}_{p_1 p_2}$$
$$\left[v_q\right]^{q_3}_{q_1 q_2} := \left[V\right]^{q_3}_{q_1 q_2}, \tag{19}$$

with $p_3, q_3 \in [0, \chi_{o_c} - 1]$, and we also obtain the normalized singular values $\mathfrak{D}$ with rank χ_{o_c}.

In the Lanczos method, $\tilde{H}$ must be applied to a vector $\tilde{\Phi}$ in the truncated Hilbert space as many times as the dimension of the Krylov subspace. The TTNOpt package calculates $\tilde{H}\tilde{\Phi}$ by applying each term of Eq. (15) individually to reduce the computational cost. Since each operator has a $\chi \times \chi$ size, the computational cost of $\tilde{H}\tilde{\Phi}$ scales as $O(C\chi^5)$, where C is an integer that depends on the number of terms. To further reduce computational cost, we adjust the order of summation of spin operators. For example, when taking $\sum_{r \in \boldsymbol{r}, r' \in \boldsymbol{r}'} J_{rr'} \tilde{S}^+_{e,r} \tilde{S}^-_{e',r'}$ with $g' > g$ where $g = |\boldsymbol{r}|, g' = |\boldsymbol{r}'|$, we first apply $\sum_{r' \in \boldsymbol{r}'} J_{rr'} \tilde{S}^-_{e',r'}$ to $\tilde{\Phi}$ and subsequently apply $\tilde{S}^+_r$ for each $r \in \boldsymbol{r}$. Additionally, TTNOpt utilizes $\left[\tilde{\Phi}\right]_{p_1 p_2 q_1 q_2} = \sum_c \left[v_p\right]^c_{p_1 p_2} \left[v_q\right]^c_{q_1 q_2} \Big/ \sqrt{\sum_{p_1 p_2 q_1 q_2 c c'} \left[v_p\right]^c_{p_1 p_2} \left[v_q\right]^c_{q_1 q_2} \left[v_p^*\right]^{c'}_{p_1 p_2} \left[v_q^*\right]^{c'}_{q_1 q_2}}$, where v_p and v_q are decided by the RSRG previously applied, as the initial state for the Lanczos method.

3.2.3. Main procedure

We first show the high-level procedure of the ground state search method of TTNOpt in Algorithm 1. Given a TTN state with the mixed canonical form described in Section 3.1 with above $\boldsymbol{v}$, $\boldsymbol{E}$, e_c, and $\mathfrak{D}$, TTNOpt updates TTN states based on the two-tensor update method within a sweep procedure as shown in Algorithm 2. Although the path of the sweep is not unique, it has to pass through all tensors in TTN states at least once during a sweep, even if TTN structures are not fixed. In TTNOpt, we implemented one variety of sweep procedures proposed in Ref. [44].

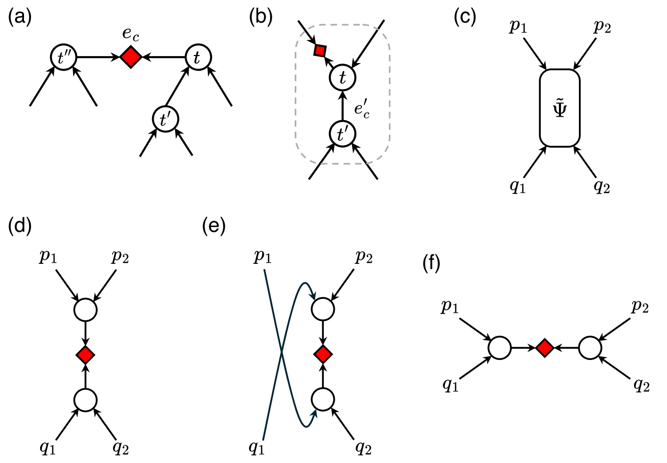

Fig. 5. Schematic diagrams of procedures of the two-site tensor update: (a) the step of selecting the next canonical center denoted as e_c' decided by using Algorithm 5, (b) the step of contraction within the next canonical region detailed in lines 12 and 13 of Algorithm 2, where a gray rounded rectangle represents the region of contracted tensors, (c) renormalized wave function $\tilde{\Psi}$ resulting from the step (b) that is updated by using the Lanczos method in the ground state search, and (d),(e), and (f) three possible candidates of SVD of $\tilde{\Psi}$ corresponding to Eq. (20).

To illustrate Algorithm 2, let us introduce the set of edge labels $\boldsymbol{e} = \{0, 1, \dots, 2N_\mathrm{t}\}$. We assign a flag $f_e \in \{0, 1\}$ to each bond $e \in \boldsymbol{e}$ to track the path and completion of the sweep. In the algorithm, flags for the bonds connecting to bare sites are initialized 1, i.e., $f_e = 1$ with $e \in [0, N-1]$. We also use the bond distances d_e from the origin bond o_c. The distance d_e is decided based on the Breadth-First Search algorithm described in Algorithm 3. Furthermore, we define two terminologies: the set of block spin operators $\tilde{\boldsymbol{S}}$ for all bonds and the set of block Hamiltonians $\tilde{\boldsymbol{H}}$ for all isometries. Both of these have been initially constructed during the RSRG procedure. In the TTNOpt package, $\tilde{\boldsymbol{S}}$ and $\tilde{\boldsymbol{H}}$ are implemented as dictionaries. Here, $\tilde{\boldsymbol{H}}$ stores the block Hamiltonians $\tilde{H}_{e_1^{(i)} e_2^{(i)}}$ with a key $e_3^{(i)}$ for $i \in [0, N_\mathrm{t}-1]$ according to the isometry v_i. Meanwhile, $\tilde{\boldsymbol{S}}$ is a nested dictionary whose elements are indexed by an edge label key $e \in \boldsymbol{e}$. Each $\tilde{\boldsymbol{S}}_e$ stores a dictionary of block spin operators $\tilde{S}_{e,r}^{(\cdot)}$ with $(\cdot) \in \{z, +\}$, where the key $r \in \boldsymbol{r}_e$ represents a set of physical spin locations associated with e.

In Algorithm 2, the sweep procedure continues until all flags of bonds incoming to the renormalized region of the current canonical center e_c are equal to 1. It is worth mentioning that in our algorithm, this situation always happens when e_c returns to the origin bond o_c and all flags, except for f_{o_c}, are equal to 1. Algorithm 4 is used to detect the edges incoming to the renormalized region whose flags are 0. From these edges, we choose the edge e_c', which will be the next canonical center, by using Algorithm 5.

Let us assume that the bond e_c' connects v_t and $v_{t'}$, as shown in Fig. 5(a), and $v_{t''}$ was updated in the previous step. The flagging process in Algorithm 2 ensures that $f_{e_\mathrm{c}} = 1$ only if all bonds in the subtree rooted at the parent tensor $v_{t''}$ have a flag of 1. To update tensors v_t and $v_{t'}$, we have to obtain the renormalized wave function $\tilde{\Psi}_{e_1^{(t)} e_2^{(t)} e_1^{(t')} e_2^{(t')}}$ by using the Lanczos method. We note that TTNOpt contracts v_t, v_t', and $\mathfrak{D}$ at e_c as described in Fig. 5(a) and (b), and this contracted tensor is used in the Lanczos method as an initial renormalized wave function. The Lanczos method requires to compose the superblock Hamiltonian $\tilde{H}_{e_1^{(t)} e_2^{(t)} e_1^{(t')} e_2^{(t')}}$ according to the new canonical region specified by e_c'. Recall that the effective Hamiltonians and the block spin operators for all bonds except those at e_c are retained in $\tilde{\boldsymbol{H}}$ and $\tilde{\boldsymbol{S}}$, respectively. This means that only $\tilde{\boldsymbol{S}}_{e_\mathrm{c}}$ and $\tilde{\boldsymbol{H}}_{e_\mathrm{c}}$ are refreshed by applying $v_{t''}$ in Eqs. (12) and (13).

Algorithm 1 Main procedure of ground state search.

```
1:  Input: $H$: the definition of the Hamiltonian, $\boldsymbol{E}$: the list of three-integer tuples representing edge labels, $\boldsymbol{e}$: the list of integers of edge labels, $o_\mathrm{c}$: the integer referring to the edge label of the origin, $\mathfrak{m}$: the number of stages of calculations, $\chi_\mathrm{init}$: the maximum bond dimension for initializing tensor, $\chi$: the maximum bond dimensions for each stage of calculations, $\boldsymbol{n}_\mathrm{max}$: the maximum number of sweeps for each stage of calculations, $\epsilon_\mathrm{E}$: the threshold for the energies, $\epsilon_{\mathcal{S}}$: the shoreshold for the EEs, $l$: the number of consecutive times TTNOpt detects the TTN state as converged before terminating the optimization. This value is set to 2 by default.
2:  function MAIN($H, \boldsymbol{E}, \boldsymbol{e}, o_\mathrm{c}, \mathfrak{m}, \chi_\mathrm{init}, \chi, \boldsymbol{n}_\mathrm{max}, \epsilon_\mathrm{E}, \epsilon_{\mathcal{S}}, l$)
3:      $\boldsymbol{v}, \mathfrak{D}, \tilde{\boldsymbol{S}}, \tilde{\boldsymbol{H}}$ := INITIALIZE_TTN ($H, \boldsymbol{E}, o_\mathrm{c}, \chi_\mathrm{init}$)        ▷ See Section 3.2.2
4:      for $m = 1$ to $\mathfrak{m}$ do
5:          $c := 0$
6:          for $n = 1$ to $n_{\mathrm{max},m}$ do
7:              $\boldsymbol{E}, \boldsymbol{v}, \mathfrak{D}, \tilde{\boldsymbol{S}}, \tilde{\boldsymbol{H}}, \mathfrak{E}, \mathfrak{S}$ :=SWEEP($\boldsymbol{E}, \boldsymbol{e}, o_\mathrm{c}, \boldsymbol{v}, \mathfrak{D}, \tilde{\boldsymbol{S}}, \tilde{\boldsymbol{H}}, \chi_m$)    ▷ $\mathfrak{E}, \mathfrak{S}$ are sets of bond energies and EEs.
8:              if $n > 1$ then
9:                  if $\boldsymbol{E}_b = \boldsymbol{E}_b'$ for $b \in [0, N_\mathrm{t}-1]$ then
10:                     if $|1 - \frac{\mathfrak{E}_b}{\mathfrak{E}_b'}| < \epsilon_\mathrm{E}$ for $b \in [N, 2N_\mathrm{t}]$ then
11:                         if $|\mathfrak{S}_b - \mathfrak{S}_b'| < \epsilon_{\mathcal{S}}$ for $b \in [0, 2N_\mathrm{t}]$ then
12:                             $c := c + 1$
13:                             if $c > l$ then
14:                                 Break
15:                             end if
16:                         end if
17:                     end if
18:                 else
19:                     $c := 0$
20:                 end if
21:             end if
22:             $\boldsymbol{E}' := \boldsymbol{E}$
23:             $\mathfrak{E}' := \mathfrak{E}$
24:             $\mathfrak{S}' := \mathfrak{S}$
25:         end for
26:     end for
27: end function
```

After obtaining $\left[\tilde{\Psi}_{e_1^{(t)} e_2^{(t)} e_1^{(t')} e_2^{(t')}}\right]_{p_1 p_2 q_1 q_2}$, where $\{p_1, p_2, q_1, q_2\}$ are indices corresponding to edge labels $\{e_1^{(t)}, e_2^{(t)}, e_1^{(t')}, e_2^{(t')}\}$, the DECOMPOSE_TENSOR function is performed to update tensors and local structure. For the reconnection of $\tilde{\Psi}_{e_1^{(t)} e_2^{(t)} e_1^{(t')} e_2^{(t')}}$, there exist three possible index orders: $(p_1 p_2 \mid q_1 q_2)$, $(p_1 q_2 \mid p_2 q_1)$, and $(p_1 q_1 \mid p_2 q_2)$ as shown in Fig. 5(d)–(f). In DECOMPOSE_TENSOR function, the EEs for all three configuration are computed by performing a full SVD, which has a computational cost of $O(\chi^6)$. The EEs are given by

$$\begin{aligned} S^{(p_1 p_2 | q_1 q_2)} &= -\sum_c (D_c)^2 \ln (D_c)^2, \\ S^{(p_1 q_2 | p_2 q_1)} &= -\sum_c (D_c')^2 \ln (D_c')^2, \\ S^{(p_1 q_1 | p_2 q_2)} &= -\sum_c (D_c'')^2 \ln (D_c'')^2, \end{aligned} \tag{20}$$

where

$$\begin{aligned} \left[\tilde{\Psi}_{e_1^{(t)} e_2^{(t)} e_1^{(t')} e_2^{(t')}}\right]_{p_1 p_2 q_1 q_2} &= \sum_c \left[U\right]_{p_1 p_2}^{c} \left[D\right]_c \left[V\right]_{q_1 q_2}^{c}, \\ &= \sum_c \left[U'\right]_{p_1 q_2}^{c} \left[D'\right]_c \left[V'\right]_{p_2 q_1}^{c}, \\ &= \sum_c \left[U''\right]_{p_1 q_1}^{c} \left[D''\right]_c \left[V''\right]_{p_2 q_2}^{c}, \end{aligned} \tag{21}$$

Algorithm 2 Sweep procedure for the ground state search.

```
1: **In/Output:** $\boldsymbol{E}, \boldsymbol{e}$: the variables introduced in Algorithm 1, $\boldsymbol{v}$: the list
   of isometric tensors, $\mathfrak{D}$ : the normalized vector containing up to $\chi$
   singular values, $\tilde{\boldsymbol{S}}$ : the set of block spin operators for all edges, and
   $\tilde{\boldsymbol{H}}$ : the set of block Hamiltonian for all isometries.
2: **Input:** $o_\mathrm{c}$: the variables introduced in Algorithm 1, and $\chi$: the maxi-
   mum bond dimension of TTN.
3: **Output:** $\mathfrak{E}$: the set of bond energies obtained by the Lanczos method,
   and $\mathfrak{S}$: the set of EEs obtained from Eq. (20).
4: **function** SWEEP($\boldsymbol{E}, \boldsymbol{e}, o_\mathrm{c}, \boldsymbol{v}, \mathfrak{D}, \tilde{\boldsymbol{S}}, \tilde{\boldsymbol{H}}, \chi$)
5:     $e_\mathrm{c} := o_\mathrm{c}$
6:     $\boldsymbol{f} := \{0 \mid f_e, \text{where } e \in \boldsymbol{e}\}$
7:     $\boldsymbol{d} :=$ SET_DISTANCE($\boldsymbol{E}, \boldsymbol{e}, e_\mathrm{c}$)
8:     $\mathfrak{E} := \{\}$                          ▷ Initialize the set of energies $\mathfrak{E}$.
9:     $\mathfrak{S} := \{\}$                             ▷ Initialize the set of EEs $\mathfrak{S}$.
10:    **while** CANDIDATE_EDGE_INDICES($\boldsymbol{E}, e_\mathrm{c}, \boldsymbol{f}$) $\neq \{\}$ **do**
11:        $e'_\mathrm{c}, t, t', t'' :=$ LOCAL_TWO_TENSOR($\boldsymbol{E}, e_\mathrm{c}, \boldsymbol{f}, \boldsymbol{d}$)
                                                           ▷ See Fig. 5(a).
12:        **if** $\left\{ f_e = 1 \mid e \in \{e_1^{(t'')}, e_2^{(t'')}\} \right\}$ **then**
                            ▷ Recall that $E_{t''} = (e_1^{(t'')}, e_2^{(t'')}, e_3^{(t'')})$.
13:            **if** $e_c \neq o_\mathrm{c}$ **then**
14:                $f_{e_\mathrm{c}} := 1$            ▷ At this point, it satisfies $e_\mathrm{c} = e_3^{(t'')}$
15:            **end if**
16:        **end if**
   ▷ If $f_{e_\mathrm{c}}$ becomes 1, it calculates expectation values according to $v_{t''}$
   with $\tilde{\Psi}$ obtained at the previous step.
17:        Update $\tilde{\boldsymbol{S}}_{e_\mathrm{c}}$ by Eq. (12) with $v_{t''}$
18:        Update $\tilde{\boldsymbol{H}}_{e_\mathrm{c}}$ by Eq. (13) with $v_{t''}$
19:        $\tilde{\Psi} := v_t \circ \mathfrak{D} \circ v_{t'}$
              ▷ $\circ$ denotes the contraction of tensors according to the same
   indices based on $\boldsymbol{E}_t$ and $\boldsymbol{E}_{t'}$. See Figs. 5 (b) and (c).
20:        $\tilde{\Psi}$, E :=LANCZOS($\tilde{\Psi}, \tilde{\boldsymbol{S}}, \tilde{\boldsymbol{H}}$)
21:        $\mathfrak{E}_{e'_\mathrm{c}} :=$ E
22:        $v_t, \mathfrak{D}, v_{t'}, \mathcal{S} :=$ DECOMPOSE_TENSOR ($\tilde{\Psi}, \chi$)
23:        $\mathfrak{S}_{e'_\mathrm{c}} := \mathcal{S}$                ▷ $\mathcal{S}$ is calculated by Eq. (20).
24:        **for** $r \in \{e_1^{(t)}, e_2^{(t)}, e_1^{(t')}, e_2^{(t')}\}$ **do**
25:            **if** $r \in [0, N-1]$ **then**
26:                $\mathcal{S}_r :=$ SITE_EE ($\tilde{\Psi}, r$)                ▷ See Fig. 6.
27:                $\mathfrak{S}_r := \mathcal{S}_r$
28:            **end if**
29:        **end for**
30:        $e_\mathrm{c} := e'_\mathrm{c}$
31:        Update $E_t, E_{t'}$
32:        $\boldsymbol{d} :=$ SET_DISTANCE($\boldsymbol{E}, \boldsymbol{e}, o_\mathrm{c}$)
33:    **end while**
         ▷ Expectation values are calculated using $v_t$ and $v_{t'}$ which
   construct $\tilde{\Psi}$ obtained at the last step.
   **return** $\boldsymbol{E}, \boldsymbol{v}, \boldsymbol{e}, \mathfrak{D}, \tilde{\boldsymbol{S}}, \tilde{\boldsymbol{H}}, \mathfrak{E}, \mathfrak{S}$
34: **end function**
```

respectively and c is up to χ^2. TTNOpt then selects the structure with the smallest EE, if users set **opt_structure.type** to 1. However, if the minimum EE $\mathcal{S}_\mathrm{min}$ and $\mathcal{S}^{(p_1p_2|q_1q_2)}$ of the original structure [Fig. 5(d)], satisfy the condition $|\mathcal{S}^{(p_1p_2|q_1q_2)} - \mathcal{S}_\mathrm{min}| < \epsilon_\mathcal{S}$, TTNOpt retains the original connection to avoid insignificant variations in the TTN structure. Here, $\epsilon_\mathcal{S}$ is defined by **numerics.entanglement_convergence_threshold**. Once the optimal structure is determined, TTNOpt truncates any singular values exceeding the rank χ set by **numerics.max_bond_dimension**, while accounting for degeneracies in the singular values with $\delta_\mathcal{S}$.

Since the reconnection procedure employed in TTNOpt is local, the solution may be trapped in local minima, especially in complex systems such as disordered ones [46,51]. To overcome this problem, TTNOpt has a function to select a structure based on relative probabilities. The heat-bath method works by evaluating the EEs for the three possible reconnections and sampling the one with a distribution given by the following expression

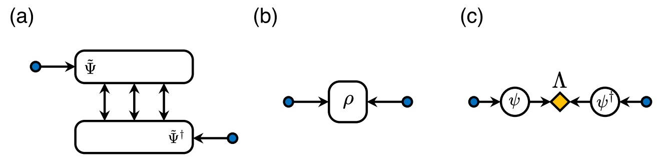

Fig. 6. Schematic diagrams of the process of the SITE_EE function. (a) and (b) represent the calculation of $\left[\rho\right]_{p_1p'_1} = \sum_{p_2q_1q_2} \left[\tilde{\Psi}\right]_{p_1p_2q_1q_2} \left[\tilde{\Psi}^*\right]_{p'_1p_2q_1q_2}$, where p_1 here is the index for a bare site s_r with $r \in [0, N-1]$. (c) describes the diagonalization for ρ to obtain the EE on the bond p_1, i.e., $\left[\rho\right]_{p_1p'_1} = \sum_c \left[\psi\right]_{p_1c} \left[\Lambda\right]_c \left[\psi^*\right]_{cp'_1}$ and $\mathcal{S} = -\sum_c \Lambda_c \ln \Lambda_c$.

$$
\begin{aligned}
&P^{(p_1p_2|q_1q_2)} \propto \exp\left[-\mathcal{S}^{(p_1p_2|q_1q_2)}/T\right] , \\
&P^{(p_1q_2|p_2q_1)} \propto \exp\left[-\mathcal{S}^{(p_1q_2|p_2q_1)}/T\right] , \\
&P^{(p_1q_1|p_2q_2)} \propto \exp\left[-\mathcal{S}^{(p_1q_1|p_2q_2)}/T\right] , \qquad (22)
\end{aligned}
$$

with

$$T = 2^{-n/n_\tau} T_0 , \qquad (23)$$

where an initial temperature T_0 and a decay factor n_τ are set by **numerics.opt_structure.temperature** and **numerics.opt_structure.tau**, respectively, and $n \in [0, n_\mathrm{max} - 1]$ represents the sweep number with n_max assigned by **numerics.max_sweep_nums**. To achieve the convergence of the TTN structure during the sweep process, TTNOpt exponentially decreases T to 0 (see Ref. [46]).

Furthermore, TTNOpt can select the structure with the minimum truncation error if **opt_structure.type** is set to 2. Here, the truncation error is defined as

$$\Delta = 1 - \sum_{c=1}^{\chi} (D_c)^2. \qquad (24)$$

for $(p_1p_2 \mid q_1q_2)$ and similarly for the other decompositions.

As shown in Algorithm 1, the RUN_SWEEP is repeated until the number of sweeps reaches n_max set by **numerics.max_num_sweeps** or the structure described by $\boldsymbol{E}$, variational energies, and EEs have been converged. TTNOpt saves the variational energy obtained by the Lanczos diagonalization performed at auxiliary bonds in the set $\mathfrak{E}$, as well as the EEs on all bonds, including physical ones, in the set $\mathfrak{S}$. These values are always overwritten for the same bonds in the sweep. To check for convergence, TTNOpt computes the difference in variatonal energies and EEs for all considered bonds with those from the previous sweep, and judges the convergence with ϵ_E and $\epsilon_\mathcal{S}$ specified by **numerics.energy_convergence_threshold** and **numerics.entanglement_convergence_threshold**, respectively.

3.2.4. *Calculating expectation values*

The TTNOpt package computes expectation values for one and two-site spin operators in sweep procedures using block spin operators $\tilde{\boldsymbol{S}}$. To ensure that the computation covers all bare sites and site pairs without duplication or omission, TTNOpt assumes the structure remains unchanged during the calculation of the expectation values. Therefore, if users specify the structural optimization conducted, TTNOpt performs an additional sweep to calculate the expectation values after the first update stage, where $m = 1$, for the TTN with the optimized structure. In the stages of $m > 1$, the same calculations are carried out in every sweep since the structure is fixed.

TTNOpt calculates expectation values using $\tilde{\Psi}_{e_1^{(t)}e_2^{(t)}e_1^{(t'')}e_2^{(t'')}}$ in Fig. 5(a) at each step. Single-site expectation values are computed when the renormalized wave function $\tilde{\Psi}$ is directly associated with the corresponding physical site. Two-site expectation values are evaluated at the

Algorithm 3 Set distances from the edge e using the Breadth-First Search.

```
Input: $\boldsymbol{E}, \boldsymbol{e}, o_c$: the variables introduced in Algorithm 1.
Output: $\boldsymbol{d}$: the list of distances from edge $o_c$
function SET_DISTANCE($\boldsymbol{E}, \boldsymbol{e}, o_c$)
    for $e \in \boldsymbol{e}$ do
        $\boldsymbol{A}_e := \bigcup_{\mathcal{E} \in \boldsymbol{E}} \{e' \mid e \in \mathcal{E}, e' \in \mathcal{E}, e' \neq e\}$
    end for
    $\boldsymbol{d} := \{0 \mid d_e, \text{where } e \in \boldsymbol{e}\}$   ▷ Initialize entries of distance list as 0
    Initialize an empty queue $Q$
    Enqueue $o_c$ into $Q$
    while $Q$ is not empty do
        Dequeue $e'$ from $Q$
        $d' := d_{e'}$
        for each neighbor edge $u$ of $e'$ in $\boldsymbol{A}_{e'}$ do
            if $d_u = 0$ then
                $d_u := d_{e'} + 1$
                Enqueue $u$ into $Q$
            end if
        end for
    end while
    return $\boldsymbol{d}$
end function
```

Algorithm 4 Detect candidate edges.

```
Input: $\boldsymbol{E}$: the valiable introduced in Algorithm 1, $e_c$: the integer of edge
    label of the current canonical center, and $\boldsymbol{f}$: the list of bools of flag
    at each edge label.
Output: $\boldsymbol{c}$: the list of edge indices
function CANDIDATE_EDGE_INDICES($\boldsymbol{E}, e_c, \boldsymbol{f}$)
    $\boldsymbol{c} := \bigcup_{(\mathcal{E}_1,\mathcal{E}_2,\mathcal{E}_3) \in \boldsymbol{E}} \{e \mid e \in \{\mathcal{E}_1, \mathcal{E}_2\}, \mathcal{E}_3 = e_c\}$   ▷ Find edges
    adjescent to canonical center.
    $\boldsymbol{c} := \{i \in \boldsymbol{c} \mid f_i = 0\}$   ▷ Filter by flag $\boldsymbol{f}$.
    return $\boldsymbol{c}$
end function
```

Algorithm 5 Select two local tensors connected by the edge with the largest distance from the initial canonical center.

```
Input: $\boldsymbol{E}, e_c, \boldsymbol{f}$: the variables introduced in Algorithm 4, and $\boldsymbol{d}$: the list
    of integers of distance at each edge
Output: $e'_c$: the integer of edge label of the next canonical center, $t, t', t''$:
    the integers of tensor labels.
function LOCAL_TWO_TENSOR($\boldsymbol{E}, e_c, \boldsymbol{f}, \boldsymbol{d}$)
    $\boldsymbol{c}$ := CANDIDATE_EDGE_INDICES($\boldsymbol{E}, e_c, \boldsymbol{f}$)
    $d_{\max} := \max(\{d_e \mid e \in \boldsymbol{c}\})$
    $\boldsymbol{c} := \{e \in \boldsymbol{c} \mid d_e = d_{\max}\}$
    $e'_c := c_1$   ▷ Select the first edge in $\boldsymbol{c}$
    for $i \in [0, N_t - 1]$ do
        if $e_c \in E_i \wedge e'_c \in E_i$ then
            $t := i$
        end if
        if $e_c \notin E_i \wedge e'_c \in E_i$ then
            $t' := i$
        end if
        if $e_c \in E_i \wedge e'_c \notin E_i$ then
            $t'' := i$
        end if
    end for
    return $(e'_c, t, t', t'')$
end function
```

step where the edge of the canonical center, e'_c, lies on the minimal path connecting the two physical sites. See Algorithm 1 for the details.

Let us denote for simplicity the renormalized wave function as $\left[\tilde{\Psi}\right]_{p_1 p_2 q_1 q_2}$ with $(p, q) = (t'', t)$ and eliminate the subscripts $\{e_1^{(t)}, e_2^{(t)}, e_1^{(t'')}, e_2^{(t'')}\}$. When v_p directly connects with the physical site r, expectation values of spin operators for the site are evaluated by the following equation

$$\langle s_r^\alpha \rangle = \begin{cases} \sum\limits_{p_1 p_2 q_1 q_2 p'_1} \left[\tilde{\Psi}^*\right]_{p_1 p_2 q_1 q_2} \left[\tilde{\Psi}\right]_{p'_1 p_2 q_1 q_2} \left[s_r^{(\alpha)}\right]_{p_1 p'_1} (r = e_1^{(p)}) \\ \sum\limits_{p_1 p_2 q_1 q_2 p'_2} \left[\tilde{\Psi}^*\right]_{p_1 p_2 q_1 q_2} \left[\tilde{\Psi}\right]_{p_1 p'_2 q_1 q_2} \left[s_r^{(\alpha)}\right]_{p_2 p'_2} (r = e_2^{(p)}) \end{cases}, \tag{25}$$

with $\alpha \in \{x, y, z\}$. The two-site correlation between sites $r \in \boldsymbol{r}_1^{(p)}$ and $r' \in \boldsymbol{r}_2^{(p)}$ are obtained by

$$\langle s_r^\alpha s_{r'}^\beta \rangle = \sum_{p_1 p_2 q_1 q_2 p'_1 p'_2} \left[\tilde{\Psi}^*\right]_{p_1 p_2 q_1 q_2} \left[\tilde{\Psi}\right]_{p'_1 p'_2 q_1 q_2} \left[\tilde{S}^{(\alpha)}_{e_1^{(p)}, r}\right]_{p_1 p'_1} \left[\tilde{S}^{(\beta)}_{e_2^{(p)}, r'}\right]_{p_2 p'_2}, \tag{26}$$

with $(\alpha, \beta) \in \{x, y, z\}^{\otimes 2}$.

After completing a sweep, TTNOpt calculates expectation values concerning the origin bond o_c with the renormalized wave function $\tilde{\Psi}_{e_1^{(t)} e_2^{(t)} e_1^{(t')} e_2^{(t')}}$ obtained at the final step of sweeps as shown in Fig. 5(b), (c). We denote $\left[\tilde{\Psi}_{e_1^{(t)} e_2^{(t)} e_1^{(t')} e_2^{(t')}}\right]_{p_1 p_2 q_1 q_2}$ as $\left[\tilde{\Psi}\right]_{p_1 p_2 q_1 q_2}$ eliminating the subscripts with $(p, q) \in \{(t, t'), (t', t)\}$. Single site expectation values are obtained using the same equation as Eq. (25), which is calculated only in the case that one (or some) of the bonds $\{e_1^{(p)}, e_2^{(p)}, e_1^{(q)}, e_2^{(q)}\}$ is connected directly with a physical site r. Regarding two-site correlations, TTNOpt calculates expectation values for all spin pairs that have not yet been evaluated during the sweeps, using the renormalized wave function associated with the pair (r, r'), where $r \in \{\boldsymbol{r}_1^{(p)}, \boldsymbol{r}_2^{(p)}\}$ and $r' \in \{\boldsymbol{r}_1^{(q)}, \boldsymbol{r}_2^{(q)}\}$. For example, in the case of $(r, r') = (\boldsymbol{r}_1^{(p)}, \boldsymbol{r}_1^{(q)})$, the corresponding contraction is evaluated as

$$\langle s_r^{(\alpha)} s_{r'}^{(\beta)} \rangle = \sum_{\substack{p_1 p_2 q_1 q_2 \\ p'_1 q'_1}} \left[\tilde{\Psi}^*\right]_{p_1 p_2 q_1 q_2} \left[\tilde{\Psi}\right]_{p'_1 p_2 q'_1 q_2} \left[\tilde{S}^{(\alpha)}_{e_1^{(p)}, r}\right]_{p_1 p'_1} \left[\tilde{S}^{(\beta)}_{e_1^{(q)}, r'}\right]_{q_1 q'_1}. \tag{27}$$

In our algorithm, the expectation values are computed at the step where the distance between the renormalized region in which the wave function $\tilde{\Psi}$ is obtained and the corresponding physical sites is minimized. This approach stems from the idea that minimizing the number of renormalization steps for spin operators can reduce the loss of accuracy induced by truncation. However, it remains an open question whether the current strategy outperforms the alternative approach in which expectation values are evaluated using the fixed TTN wavefunction $|\Psi\rangle$ after the completion of the sweep.

3.3. Factorizing tensors

Let us assume that a rank-N tensor $\Psi_{s_0,\ldots,s_{N-1}}$ is given by **target.tensor**, and Ψ is normalized as

$$\Psi := \frac{\Psi}{\sqrt{\sum\limits_{s_0,\ldots,s_{N-1}} \left[\Psi\right]_{s_0,\ldots,s_{N-1}} \left[\Psi^*\right]_{s_0,\ldots,s_{N-1}}}}, \tag{28}$$

which is performed by TTNOpt itself before the factorization. TTNOpt then proceeds to decompose Ψ by using the sequential SVD into the MPN form [29] with a bond dimension χ_{init} specified in **numerics.initial_bond_dimension** [Fig. 3(c)]. The tensor is successively factorized by SVDs from both ends to the center of the MPN. Importantly, in this process, the SVDs are applied according to the original index order. If users set **numerics.opt_structure.type** as 1 or 2, TTNOpt runs sweeps with reconnection of local structures [Fig. 3(d)], starting from the initial MPN prepared as above. During these sweeps, TTNOpt applies SVDs with χ_{init} to the renormalized wave function $\tilde{\Psi}$, which is obtained by contracting two tensors as shown in Fig. 3(b). TTNOpt then selects the structure with the minimum EE from Fig. 5(d)–(f). This process continues until the TTN structure is fixed and the EEs are converged within the value ϵ_S specified by **numerics.entanglement_convergence_threshold**.

If users specify variables of **numerics.fidelity**, TTNOpt further optimizes the TTN state based on the fidelity with the original tensor Ψ. To obtain the optimal renormalized wave function $\tilde{\Psi}_{e_1^{(p)}e_2^{(p)}e_1^{(q)}e_2^{(q)}}$, it is necessary to compute the environment $\mathcal{E}_{e_1^{(p)}e_2^{(p)}e_1^{(q)}e_2^{(q)}}$ associated with two tensors v_p and v_q within the canonical region:

$$\mathcal{E}_{e_1^{(p)}e_2^{(p)}e_1^{(q)}e_2^{(q)}} = \Psi \prod_{i\in[0,N_t-1]/\{p,q\}} \circ\, v_i^*, \tag{29}$$

where $\circ$ represents the contraction between tensors with the same indices following $\boldsymbol{E}$. We directly embed the environment as

$$\left[\tilde{\Psi}\right]_{p_1p_2q_1q_2} := \frac{\left[\mathcal{E}\right]_{p_1p_2q_1q_2}}{\sqrt{\sum_{p_1p_2q_1q_2}\left[\mathcal{E}\right]_{p_1p_2q_1q_2}\left[\mathcal{E}^*\right]_{p_1p_2q_1q_2}}}, \tag{30}$$

where we omitted the subscripts on tensors, to locally maximize the fidelity with Ψ. The contraction of $\mathcal{E}$ is performed from the physical sites to the canonical center, which ensures that the environment tensor is built step by step while respecting the structure of the network.

TTNOpt performs SVD on $\tilde{\Psi}$ [Eq. (30)] and updates TTN using the obtained tensors, incorporating structural optimization as specified in the input file. TTNOpt iterates the sweeps until the TTN state converges with respect to EEs, fidelity, and network structure. The convergence criteria of EE and fidelity are set by the thresholds ϵ_S and ϵ_F, specified in **numerics.fidelity.convergence_entanglement** and **numerics.fidelity.convergence_threshold**, respectively.

3.4. Reconstructing TTNs

If the TTN state is loaded by **target.tensors**, users must specify **numerics.max_sweep_num** and **numerics.opt_structure.type** as either 1 or 2. TTNOpt applies sweeps to the TTN to reconstruct its network by using DECOMPOSE_TENSOR in the same way as described in the first paragraph of Section 3.3 [Fig. 3(d)]. It is emphasized that TTNOpt retains up to χ_{init} singular values during the SVD, where χ_{init} is the maximum bond dimension of the given TTN state. The calculation terminates when the TTN state has converged concerning both the network structure and the EE, or when the maximum number of sweeps is reached.

4. Benchmark results

4.1. Hierarchical chain model

To briefly demonstrate the TTNOpt package, we performed the ground state search for the $S = 1/2$ hierarchical chain model [44] with system size $N = 2^d$ of an integer d, defined as

$$H = \sum_{h=0}^{d-1} \sum_{i\in I(h)} J\alpha^h \boldsymbol{s}_i \cdot \boldsymbol{s}_{i+1}, \tag{31}$$

where $J > 0$ is the base coupling constant, and $0 < \alpha \leq 1.0$ is the decay factor for the coupling strength. In Eq. (31), an integer $h \in [0, d-1]$ represents the height in the perfect binary tree (PBT) structure, and an integer set $I(h) = \left\{i \mid i = 2^h(2k+1) - 1,\ \text{where } k = 0, 1, \ldots, 2^{d-h-1} - 1\right\}$ specifies pairs of adjacent sites $(i, i+1)$ according to the PBT structure, as illustrated in Fig. 7.

We examined the model with $(J, \alpha) = (1.0, 0.5)$ and $(1.0, 1.0)$, where the system size is $N = 256$, i.e., $d = 8$. These settings serve as reasonable litmus tests since α controls the interaction strength between adjacent spins, directly influencing the entanglement structure of the ground states. It allows for predicting ideal TTN structures: for sufficiently small

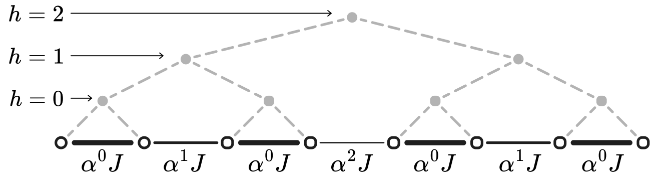

Fig. 7. Representation of the hierarchical chain model with $N = 8$. The physical sites are ordered from left to right with indices $i = 0, \ldots, N-1$.

Table 1
Maximum and average of the bond EEs in the optimized TTN and MPN when $N = 256$ with $\alpha = 0.5$ and 1.0. The EEs on the physical bonds are excluded from the analysis.

Type	Maximum	Average
optimized TTN ($\alpha = 0.5$)	0.1110	0.0618
MPN ($\alpha = 0.5$)	0.6935	0.3730
optimized TTN ($\alpha = 1.0$)	0.9977	0.9065
MPN ($\alpha = 1.0$)	1.0150	0.9376

α, by the perturbative RSRG approach [49,53], the optimal TTN structure for the ground state would be the PBT. For $\alpha = 1.0$, the system is the uniform Heisenberg chain under the open boundary conditions, and the optimal TTN structure for the ground state is expected to be an MPN-like one consisting of the dimer units as shown in Ref. [44].

We ran the variational algorithm with the maximum bond dimension $\chi = 20$ to find the optimal TTN structures. This choice of χ provides sufficient accuracy for our discussion, as the truncation errors have been nearly 1.0×10^{-15} when $\alpha = 0.5$, and have remained below 1.0×10^{-6} when $\alpha = 1.0$. Additionally, we set the maximum number of sweeps to 50 and performed the calculation with $\delta_{\mathrm{E}} = 1 \times 10^{-11}$ and $\delta_S = 1 \times 10^{-10}$, respectively. The remaining parameters used in the calculation can be found in the input file "samples/ground_state_search/hierarchical".

We demonstrate that, in either case of $\alpha = 0.5$ or 1.0, TTNOpt can achieve the same optimal structures as Ref. [44], where the system up to $N = 128$ sites was treated. Furthermore, to show the importance of TTN structures, we present the bond EEs in the optimized TTN and MPN in Table 1. It shows that the optimal TTNs reduced both the maximum and average EEs compared to the case with MPN. The reduction is significant, especially in the case of $\alpha = 0.5$. On the other hand, in $\alpha = 1.0$, it is subtle since the MPN and the dimer MPN are similar structurally.

4.2. Multivariable quantics function

Recently, the impact of TTN structures on approximating the general tensor data and compressing quantics tensors has been studied [54]. Here, we apply TTNOpt to the compression of the three-variable quantics function employed in Ref. [54] that is written as

$$f(\boldsymbol{x}) = \sum_{j=1}^{n} \cos\left(j\boldsymbol{k}_j \cdot \boldsymbol{x}\right), \tag{32}$$

where $n = 30$, $\boldsymbol{x} = (x_1, x_2, x_3) \in [0,1)^{\otimes 3}$ and $\boldsymbol{k}_j = (k_{j,1}, k_{j,2}, k_{j,3})$ with $k_j^\alpha \in \mathcal{N}(0,1)$ that is the standard normal distribution. In quantics formulation, the continous variable $x_i \in [0,1)$ with the L-bit precision is described as

$$x_i = \sum_{l=1}^{L} \frac{x_{i,l}}{2^l}, \tag{33}$$

where $x_{i,l} \in \{0,1\}$. It allows the expression of a continuous function f of m variables as a 2^{mL} dimensional tensor. In this paper, we set $L = 8$ for all three variables $m = 3$ to represent Eq. (32). The function's heatmap concerning $(x_1, x_2) \in [0,1)^{\otimes 2}$ with $x_3 = 0.5$ is depicted in Fig. 8 (a). To construct the tensor Ψ of Eq. (32), we employed a one-dimensional variable ordering, in the same order as the MPN structure shown in Fig. 8(b) that was generated via the sequential SVD from Ψ. The authors of Ref. [54]

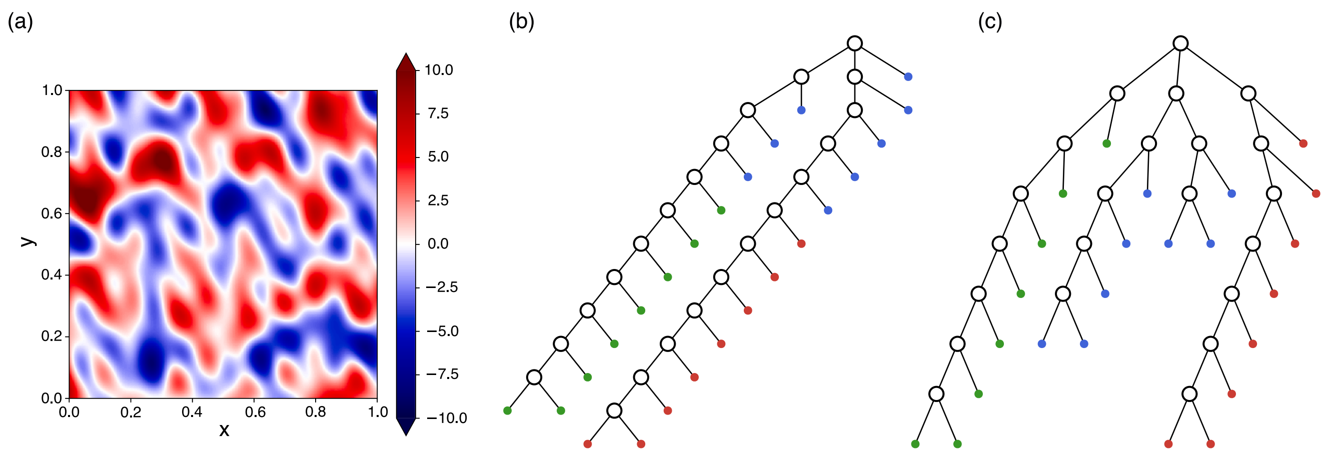

Fig. 8. (a) The heatmap of the function Eq. (32) at $x_3 = 0.5$, where $(x, y) := (x_1, x_2)$. (b) MPN decomposition of the quantics function Eq. (32) where the red, blue, and green circles distinguish variants x_1, x_2, and x_3, respectively. (c) TTN structure obtained by the two-site update algorithm in TTNOpt, which optimizes the structure while maximizing the fidelity with respect to the original functional tensor, starting from the MPN depicted in (b). In both (b) and (c), we omit the drawing of the canonical center and arrows for clarity. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

(a)

$$\begin{pmatrix}
\rho^0 & \rho^6 & \rho^6 & \rho^7 & \rho^6 & \rho^4 & \rho^5 & \rho^5 & \rho^9 & \rho^5 & \rho^6 & \rho^6 & \rho^9 & \rho^3 & \rho^8 & \rho^3 \\
\rho^6 & \rho^0 & \rho^6 & \rho^7 & \rho^6 & \rho^6 & \rho^7 & \rho^7 & \rho^9 & \rho^3 & \rho^2 & \rho^6 & \rho^9 & \rho^7 & \rho^8 & \rho^7 \\
\rho^6 & \rho^6 & \rho^0 & \rho^3 & \rho^4 & \rho^6 & \rho^7 & \rho^7 & \rho^5 & \rho^5 & \rho^6 & \rho^4 & \rho^5 & \rho^7 & \rho^4 & \rho^7 \\
\rho^7 & \rho^7 & \rho^3 & \rho^0 & \rho^5 & \rho^7 & \rho^8 & \rho^8 & \rho^4 & \rho^6 & \rho^7 & \rho^5 & \rho^4 & \rho^8 & \rho^3 & \rho^8 \\
\rho^6 & \rho^6 & \rho^4 & \rho^5 & \rho^0 & \rho^6 & \rho^7 & \rho^7 & \rho^7 & \rho^5 & \rho^6 & \rho^2 & \rho^7 & \rho^7 & \rho^6 & \rho^7 \\
\rho^4 & \rho^6 & \rho^6 & \rho^7 & \rho^6 & \rho^0 & \rho^3 & \rho^3 & \rho^9 & \rho^5 & \rho^6 & \rho^6 & \rho^9 & \rho^5 & \rho^8 & \rho^5 \\
\rho^5 & \rho^7 & \rho^7 & \rho^8 & \rho^7 & \rho^3 & \rho^0 & \rho^2 & \rho^{10} & \rho^6 & \rho^7 & \rho^7 & \rho^{10} & \rho^6 & \rho^9 & \rho^6 \\
\rho^5 & \rho^7 & \rho^7 & \rho^8 & \rho^7 & \rho^3 & \rho^2 & \rho^0 & \rho^{10} & \rho^6 & \rho^7 & \rho^7 & \rho^{10} & \rho^6 & \rho^9 & \rho^6 \\
\rho^9 & \rho^9 & \rho^5 & \rho^4 & \rho^7 & \rho^9 & \rho^{10} & \rho^{10} & \rho^0 & \rho^8 & \rho^9 & \rho^7 & \rho^2 & \rho^{10} & \rho^3 & \rho^{10} \\
\rho^5 & \rho^3 & \rho^5 & \rho^6 & \rho^5 & \rho^5 & \rho^6 & \rho^6 & \rho^8 & \rho^0 & \rho^3 & \rho^5 & \rho^8 & \rho^6 & \rho^7 & \rho^6 \\
\rho^6 & \rho^2 & \rho^6 & \rho^7 & \rho^6 & \rho^6 & \rho^7 & \rho^7 & \rho^9 & \rho^3 & \rho^0 & \rho^6 & \rho^9 & \rho^7 & \rho^8 & \rho^7 \\
\rho^6 & \rho^6 & \rho^4 & \rho^5 & \rho^2 & \rho^6 & \rho^7 & \rho^7 & \rho^7 & \rho^5 & \rho^6 & \rho^0 & \rho^7 & \rho^7 & \rho^6 & \rho^7 \\
\rho^9 & \rho^9 & \rho^5 & \rho^4 & \rho^7 & \rho^9 & \rho^{10} & \rho^{10} & \rho^2 & \rho^8 & \rho^9 & \rho^7 & \rho^0 & \rho^{10} & \rho^3 & \rho^{10} \\
\rho^3 & \rho^7 & \rho^7 & \rho^8 & \rho^7 & \rho^5 & \rho^6 & \rho^6 & \rho^{10} & \rho^6 & \rho^7 & \rho^7 & \rho^{10} & \rho^0 & \rho^9 & \rho^2 \\
\rho^8 & \rho^8 & \rho^4 & \rho^3 & \rho^6 & \rho^8 & \rho^9 & \rho^9 & \rho^3 & \rho^7 & \rho^8 & \rho^6 & \rho^3 & \rho^9 & \rho^0 & \rho^9 \\
\rho^3 & \rho^7 & \rho^7 & \rho^8 & \rho^7 & \rho^5 & \rho^6 & \rho^6 & \rho^{10} & \rho^6 & \rho^7 & \rho^7 & \rho^{10} & \rho^2 & \rho^9 & \rho^0
\end{pmatrix}$$

(b)

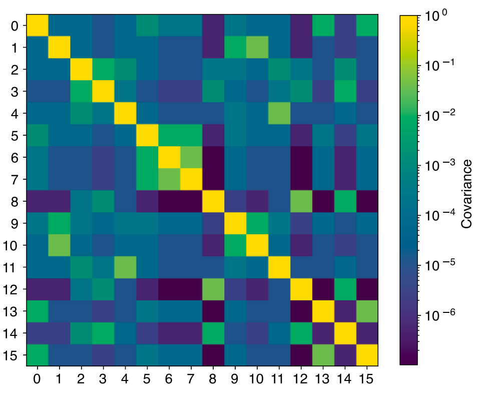

Fig. 9. The definition of the covariance matrix K in this experiment with $\rho = 0.2$. (b) The heatmap of the matrix elements of K is shown as a reference.

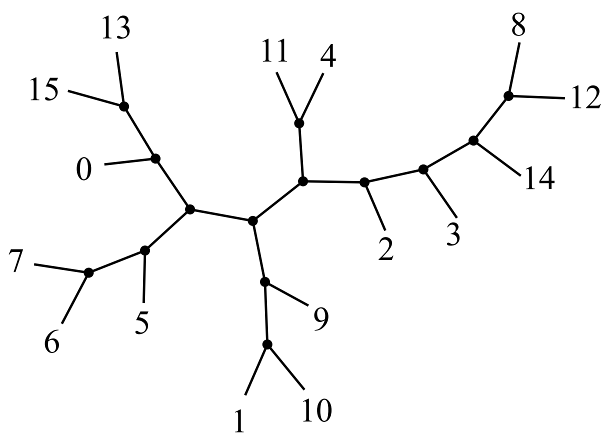

Fig. 10. The tree structure to define the covariance matrix K described in Fig. 9 in the numerical experiment.

have shown that TTN structures designed to separate each of the three variables (x_1, x_2, x_3) are reasonable, as Eq. (32) exhibits only weak correlations among them.

In our demonstration, we constructed an MPN representation of the normalized function in Eq. (32) using sequential SVD with $\chi_{\text{init}} = 4$, as depicted in Fig. 8(b). The resulting MPN was then passed into the TTN update methods of maximizing the fidelity with $[\chi_1, \chi_2, \chi_3] = [4, 8, 16]$. The structural optimization was applied to MPN when $\chi_1 = 4$. The thresholds for the convergence of fidelity and EE were set to $\epsilon_F = 1 \times 10^{-10}$ and $\epsilon_S = 1 \times 10^{-14}$, respectively.

The TTN structure obtained by TTNOpt is illustrated in Fig. 8(c). TTNOpt successfully identified a TTN structure that reflects the insight that the three variables are not correlated in a bit-wise way. Table 2 shows the EEs and fidelities obtained for both the MPN and the optimized TTN. The EEs for the TTN are lower than those from the MPN at each bond dimension. For instance, at $\chi = 16$, the EE of the MPN reaches 0.9745, while that of the TTN remains at 0.6169. It should also be noted that, when comparing the fidelities of the resulting TTN and the MPN, the memory footprint of the former is bigger than that of the latter. It

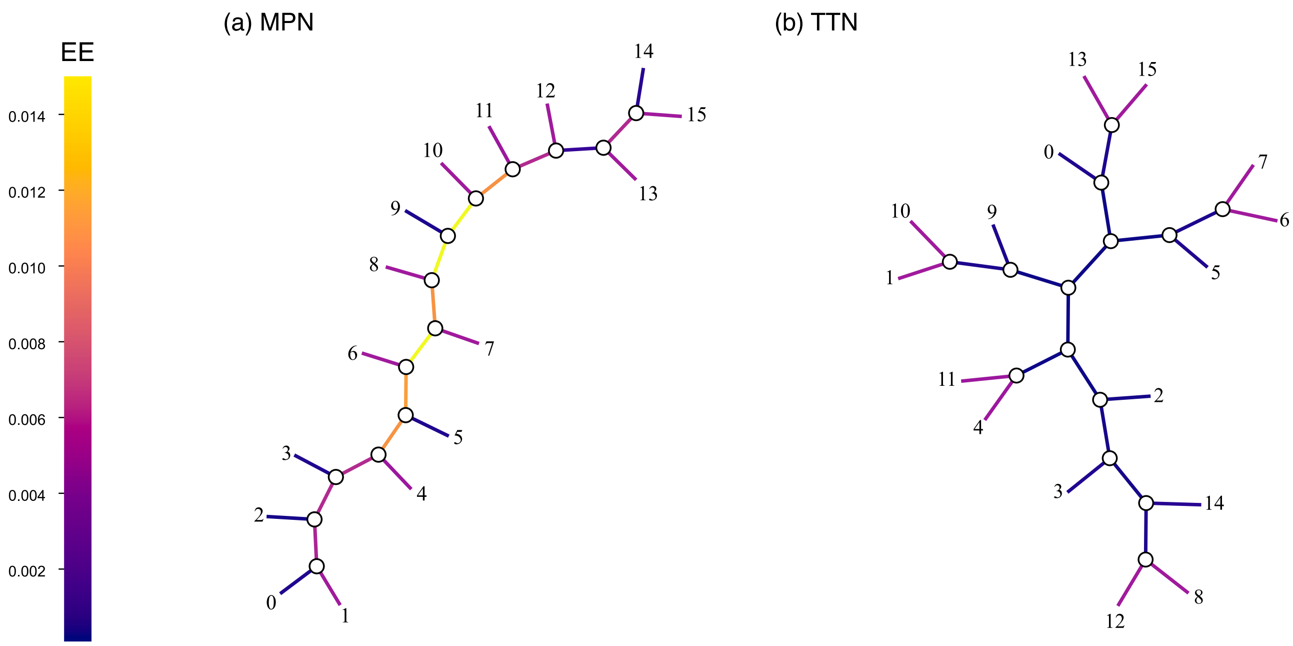

Fig. 11. The initial and final structure of the tree tensor network in the reconstruction. The colors in the edges represent the amount of entanglement entropy.

Table 2
The average bond EEs $\mathcal{S}$ and fidelities $F = |\Psi \prod_{v \in \mathcal{V}} \circ v^*|$ obtained for the optimized TTN and the fixed MPN, respectively. The average EEs are taken over auxiliary and physical bonds and rounded to four decimal places. On the other hand, the average fidelity is taken over auxiliary bonds and truncated after the fifth significant figure.

Type	TTN	MPN
$\mathcal{S}$ ($\chi = 4$)	0.3825	0.5631
$\mathcal{S}$ ($\chi = 8$)	0.5378	0.7920
$\mathcal{S}$ ($\chi = 16$)	0.6169	0.9745
F ($\chi = 4$)	0.39408	0.33479
F ($\chi = 8$)	0.72512	0.57523
F ($\chi = 16$)	0.99997	0.83884

should also be noted that, concerning the fidelities of the two TNs, the memory footprint of the resulting TTN exceeds that of the MPN. For example, at $\chi = 16$, the TTN requires approximately 1.65 times more memory. These results highlight that TTNOpt not only improves compression fidelity but also reveals efficient data sparsity based on the entanglement structure, offering a significant advantage over conventional MPN-based approaches.

4.3. Multivariate normal distribution

Finally, as an illustrative example of TTN reconstruction, we consider the TTN representation of a multivariate normal distribution [55]. The probability density function of a multivariate normal distribution with mean zero and covariance matrix K is given as follows:

$$f(\boldsymbol{x}) = \exp\left(-\frac{1}{2}\boldsymbol{x}^T K^{-1} \boldsymbol{x}\right), \tag{34}$$

where $\boldsymbol{x}$ is a D-dimensional vector and the normalization term is omitted for simplicity. In our demonstration, we set $D = 16$ and each component of $\boldsymbol{x}$ is discretized over the range $[-5, 5]$ using $L = 4$ bits of precision. We use the covariance matrix K shown in Fig. 9, which is constructed based on the tree structure illustrated in Fig. 10. The elements of K are defined using a parameter $\rho = 0.2$ and decay exponentially with the shortest path length between the variables on the tree. In this setting, $f(\boldsymbol{x})$ can be efficiently represented using a TTN whose geometry corresponds to the tree graph in Fig. 10.

Building upon this setup, we explore whether automatic structural optimization can recover the optimal network structure of the multivariate normal distribution with tree-like correlation, following the same task as in [55]. Each bare bond carries 2^4 degrees of freedom representing L-bit precision. Initially, $f(\boldsymbol{x})$ is constructed as an MPN with auxiliary bond dimension $\chi = 16$ by using the Tensor Cross Interpolation (TCI) method [56–58]. Subsequently, the reconstruction algorithm is applied, optimizing the TTN structure based on bipartite EEs. During the sweeps, we use the default values in ϵ_{EE} and δ_{EE}.

The resulting TTN is shown in Fig. 11. The optimized structure perfectly matches the tree structure in Fig. 10, successfully capturing the underlying correlations. Moreover, the bipartite EEs on the edges are reduced compared to those in the initial MPN. These results demonstrate that the TTNOpt can detect hidden correlation structures of $f(\boldsymbol{x})$ and replace the MPN with the more efficient TTN representation.

5. Summary

We have developed a TTN manipulation package for analyzing the ground states of quantum spin systems and general tensor data. The TTNOpt package conducts the local structural reconnection during the sweep procedures based on two-tensor updates. This enables us to search for effective and efficient TTN representations, surpassing the simple MPN structure.

As a demonstration, we first applied the ground state search, including the structural optimization, to the hierarchical chain model [44]. We confirmed the resulting TTN and those of EEs are consistent with those in Ref. [44]. We applied the fidelity-based update method to quantic MPN, representing the three-variable function in Ref. [54], in both cases, with and without structural optimization. We corroborated that a well-structured TTN can achieve better convergence in terms of fidelity when approximating the target data. We lastly applied TTNOpt for MPN, representing the multi-variable probability density function where the covariance matrix is explicitly defined by the tree structure [55]. As a result, the optimized TTN structure could reproduce the same structure in the covariance relation tree, and we observed a decrease in EEs compared to the MPN. Here, we note the importance of a comprehensive performance comparison between the optimized TTN and alternative TTN architectures, although our benchmarks were limited to an MPN. For a detailed study on this topic, we refer the reader to Ref. [46].

The prospects of TTNOpt are extending its scope to fermionic systems and then applying it to quantum chemistry problems. Although molecular systems are not one-dimensional, they have been analyzed using MPN-based DMRG in many cases. On the other hand, the potential benefits of introducing TTN, depending on molecular structures, especially in dendritic ones, have also been discussed [59,60]. Combining our method for time evolution algorithm such as the time-dependent variational principle (TDVP) [61,62] could also allow for more extended time evolutions by maintaining structures with low entanglement [63]. It would be implemented by alternatively applying the structural search sweeps and short-time evolution.

The factorizing tensor method is used not only to reveal the entanglement structure of the target data but also, of course, to compress the data into the TTNs. In particular, if we obtain TTN states that approximate the target quantum states, they can be converted into quantum circuits. In this scenario, a TTN structure with smaller bond dimensions would directly reduce the circuit depth [64]. It would be suitable for enhancing the usability of intermediate-sized quantum circuits [65].

The TCI algorithm [56,57] constructs the MPN that approximates the function $f(x)$ by accessing the sufficiently large number of input-output pairs $(\boldsymbol{x}, f(\boldsymbol{x}))$, rather than explicitly constructing the high-rank tensor Ψ representing $f(\boldsymbol{x})$. In the TCI, MPN has been used so far, while the use of TTN opens up the possibility of more efficient data representation. The extension of TCI to TTN is feasible [54]. However, the integration of TTN structural optimization into the TCI framework remains a subject for future research.

CRediT authorship contribution statement

Ryo Watanabe: Writing – original draft, Visualization, Software, Resources, Formal analysis, Data curation; **Hidetaka Manabe:** Writing – review & editing, Resources, Data curation; **Toshiya Hikihara:** Writing – review & editing, Validation, Supervision, Conceptualization; **Hiroshi Ueda:** Writing – review & editing, Supervision, Conceptualization.

Data availability

I have shared the link to code/data at the article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is partially supported by, KAKENHI Grant Numbers JP24K06881, JP22H01171, JP21H04446, JP21H05182, JP21H05191, and JP25KJ1773 from JSPS of Japan. We also acknowledge support from MEXT Q-LEAP Grant No. JPMXS0120319794, and from JST COI-NEXT No. JPMJPF2014, ASPIRE No. JPMJAP2319, and CREST No.JPMJCR24I1 and JPMJCR24I3. We thank K. Okunishi for insightful discussions and comments. R.W. would like to thank D. Sels, H. Shinaoka, and the Tensor Network team at the Flatiron Institute, especially J. Tindall and M. Stoudenmire, for their useful discussions. R.W. was supported by the Σ Doctoral Futures Research Grant Program from Osaka University. H.M was supported by JSTCOI-NEXT program Grant Numbers JPMJPF2014. H.U. was supported by the COE research grant in computational science from Hyogo Prefecture and Kobe City through Foundation for Computational Science.

References

[1] TTNOpt, 2025, (https://github.com/Ryo-wtnb11/TTNOpt).
[2] C. Roberts, A. Milsted, M. Ganahl, A. Zalcman, B. Fontaine, Y. Zou, J. Hidary, G. Vidal, S. Leichenauer, 2019. arxiv:1905.01330
[3] R. Orús, A practical introduction to tensor networks: matrix product states and projected entangled pair states, Ann. Phys. 349 (2014) 117–158. https://doi.org/10.1016/j.aop.2014.06.013
[4] K. Okunishi, T. Nishino, H. Ueda, Developments in the tensor network — from statistical mechanics to quantum entanglement, J. Phys. Soc. Jpn. 91 (6) (2022) 062001. https://doi.org/10.7566/JPSJ.91.062001
[5] R. Orús, Tensor networks for complex quantum systems, Nat. Rev. Phys. 1 (9) (2019) 538–550. https://doi.org/10.1038/s42254-019-0086-7
[6] S. Lu, M. Kanász-Nagy, I. Kukuljan, J.I. Cirac, et al., 2024. arXiv:2103.06872. https://doi.org/10.48550/arXiv.2103.06872
[7] B. Jobst, K. Shen, C.A. Riofrío, E. Shishenina, F. Pollmann, et al., Efficient MPS representations and quantum circuits from the fourier modes of classical image data, Quantum 8 (2024) 1544. https://doi.org/10.22331/q-2024-12-03-1544
[8] E.M. Stoudenmire, D.J. Schwab, 2017, arXiv:1605.05775. https://doi.org/10.48550/arXiv.1605.05775
[9] Z.-Y. Han, J. Wang, H. Fan, L. Wang, P. Zhang, Unsupervised generative modeling using matrix product states, Phys. Rev. X 8 (2018) 031012. https://doi.org/10.1103/PhysRevX.8.031012
[10] S. Cheng, L. Wang, T. Xiang, P. Zhang, Tree tensor networks for generative modeling, Phys. Rev. B 99 (2019) 155131. https://doi.org/10.1103/PhysRevB.99.155131
[11] I.V. Oseledets, Approximation of matrices with logarithmic number of parameters, Dokl. Math. 80 (2) (2009) 653–654. https://doi.org/10.1134/S1064562409050056
[12] B.N. Khoromskij, O(Dlog N)-quantics approximation of N-d tensors in high-dimensional numerical modeling, Constr. Approx. 34 (2) (2011) 257–280. https://doi.org/10.1007/s00365-011-9131-1
[13] T. Nishino, Y. Hieida, K. Okunishi, N. Maeshima, Y. Akutsu, A. Gendiar, Two-dimensional tensor product variational formulation, Prog. Theor. Phys. 105 (3) (2001) 409–417. https://doi.org/10.1143/PTP.105.409
[14] G. Vidal, Entanglement renormalization, Phys. Rev. Lett. 99 (2007) 220405. https://doi.org/10.1103/PhysRevLett.99.220405
[15] J. Jordan, R. Orús, G. Vidal, F. Verstraete, J.I. Cirac, Classical simulation of infinite-size quantum lattice systems in two spatial dimensions, Phys. Rev. Lett. 101 (2008) 250602. https://doi.org/10.1103/PhysRevLett.101.250602
[16] J.I. Cirac, D. Pérez-García, N. Schuch, F. Verstraete, Matrix product states and projected entangled pair states: concepts, symmetries, theorems, Rev. Mod. Phys. 93 (2021) 045003. https://doi.org/10.1103/RevModPhys.93.045003
[17] G. Vidal, Efficient classical simulation of slightly entangled quantum computations, Phys. Rev. Lett. 91 (2003) 147902. https://doi.org/10.1103/PhysRevLett.91.147902
[18] G. Vidal, Efficient simulation of one-dimensional quantum many-body systems, Phys. Rev. Lett. 93 (2004) 040502. https://doi.org/10.1103/PhysRevLett.93.040502
[19] Y.-Y. Shi, L.-M. Duan, G. Vidal, Classical simulation of quantum many-body systems with a tree tensor network, Phys. Rev. A 74 (2006) 022320. https://doi.org/10.1103/PhysRevA.74.022320
[20] L. Tagliacozzo, G. Evenbly, G. Vidal, Simulation of two-dimensional quantum systems using a tree tensor network that exploits the entropic area law, Phys. Rev. B 80 (2009) 235127. https://doi.org/10.1103/PhysRevB.80.235127
[21] V. Murg, F. Verstraete, O. Legeza, R.M. Noack, Simulating strongly correlated quantum systems with tree tensor networks, Phys. Rev. B 82 (2010) 205105. https://doi.org/10.1103/PhysRevB.82.205105
[22] G. Evenbly, G. Vidal, Algorithms for entanglement renormalization, Phys. Rev. B 79 (2009) 144108. https://doi.org/10.1103/PhysRevB.79.144108
[23] M.P. Zaletel, F. Pollmann, Isometric tensor network states in two dimensions, Phys. Rev. Lett. 124 (2020) 037201. https://doi.org/10.1103/PhysRevLett.124.037201
[24] D. Sauerwein, A. Molnar, J.I. Cirac, B. Kraus, Matrix product states: entanglement, symmetries, and state transformations, Phys. Rev. Lett. 123 (2019) 170504. https://doi.org/10.1103/PhysRevLett.123.170504
[25] D. Perez-Garcia, F. Verstraete, M.M. Wolf, J.I. Cirac, Matrix product state representations, Quantum Inf. Comput. 7 (5) (2007) 401–430. https://dl.acm.org/doi/10.5555/2011832.2011833.
[26] I.V. Oseledets, Tensor-train decomposition, SIAM J. Sci. Comput. 33 (5) (2011) 2295–2317. https://doi.org/10.1137/090752286
[27] S.R. White, Density matrix formulation for quantum renormalization groups, Phys. Rev. Lett. 69 (19) (1992) 2863–2866. https://doi.org/10.1103/PhysRevLett.69.2863
[28] S.R. White, Density-matrix algorithms for quantum renormalization groups, Phys. Rev. B 48 (14) (1993) 10345–10356. https://doi.org/10.1103/PhysRevB.48.10345
[29] U. Schollwöck, The density-matrix renormalization group in the age of matrix product states, Ann. Phys. 326 (1) (2011) 96–192. January 2011 Special Issue. https://doi.org/10.1016/j.aop.2010.09.012
[30] X. Xiao-Qiang, L. Wu-Ming, Generation of long-distance entanglement in spin system, Chin. Phys. Lett. 25 (7) (2008) 2346. https://doi.org/10.1088/0256-307X/25/7/005
[31] G. Ramírez, J. Rodríguez-Laguna, G. Sierra, From conformal to volume law for the entanglement entropy in exponentially deformed critical spin 1/2 chains, J. Stat. Mech. Theory Exp. 2014 (10) (2014) P10004. https://doi.org/10.1088/1742-5468/2014/10/P10004
[32] A.M. Goldsborough, R.A. Römer, Self-assembling tensor networks and holography in disordered spin chains, Phys. Rev. B 89 (2014) 214203. https://doi.org/10.1103/PhysRevB.89.214203

[33] K. Okunishi, H. Ueda, T. Nishino, Entanglement bipartitioning and tree tensor networks, Prog. Theor. Exp. Phys. 2023 (2) (2023) 023A02. https://doi.org/10.1093/ptep/ptad018

[34] K. Seki, T. Hikihara, K. Okunishi, Tensor-network strong-disorder renormalization groups for random quantum spin systems in two dimensions, Phys. Rev. B 102 (2020) 144439. https://doi.org/10.1103/PhysRevB.102.144439

[35] K. Seki, T. Hikihara, K. Okunishi, Entanglement-based tensor-network strong-disorder renormalization group, Phys. Rev. B 104 (2021) 134405. https://doi.org/10.1103/PhysRevB.104.134405

[36] G. Ferrari, G. Magnifico, S. Montangero, Adaptive-weighted tree tensor networks for disordered quantum many-body systems, Phys. Rev. B 105 (2022) 214201. https://doi.org/10.1103/PhysRevB.105.214201

[37] G.K.-L. Chan, M. Head-Gordon, Highly correlated calculations with a polynomial cost algorithm: a study of the density matrix renormalization group, J. Chem. Phys. 116 (11) (2002) 4462–4476. https://doi.org/10.1063/1.1449459

[38] O. Legeza, J. Sólyom, Optimizing the density-matrix renormalization group method using quantum information entropy, Phys. Rev. B 68 (2003) 195116. https://doi.org/10.1103/PhysRevB.68.195116

[39] G. Moritz, B.A. Hess, M. Reiher, Convergence behavior of the density-matrix renormalization group algorithm for optimized orbital orderings, J. Chem. Phys. 122 (2) (2005) 024107. https://doi.org/10.1063/1.1824891

[40] O. Legeza, L. Veis, A. Poves, J. Dukelsky, Advanced density matrix renormalization group method for nuclear structure calculations, Phys. Rev. C 92 (2015) 051303. https://doi.org/10.1103/PhysRevC.92.051303

[41] W. Li, J. Ren, H. Yang, Z. Shuai, On the fly swapping algorithm for ordering of degrees of freedom in density matrix renormalization group, J. Phys. Condens. Matter 34 (25) (2022) 254003. https://doi.org/10.1088/1361-648X/ac640e

[42] V. Murg, F. Verstraete, R. Schneider, P.R. Nagy, Ö. Legeza, Tree tensor network state with variable tensor order: an efficient multireference method for strongly correlated systems, J. Chem. Theory Comput. 11 (3) (2015) 1027–1036. PMID: 25844072, https://doi.org/10.1021/ct501187j

[43] H.R. Larsson, Computing vibrational eigenstates with tree tensor network states (TTNS), J. Chem. Phys. 151 (20) (2019) 204102. https://doi.org/10.1063/1.5130390

[44] T. Hikihara, H. Ueda, K. Okunishi, K. Harada, T. Nishino, et al., Automatic structural optimization of tree tensor networks, Phys. Rev. Res. 5 (1) (2023) 013031. https://doi.org/10.1103/PhysRevResearch.5.013031

[45] T. Hikihara, H. Ueda, K. Okunishi, K. Harada, T. Nishino, et al., 2024. arXiv:2401.16000. https://doi.org/10.48550/arXiv.2401.16000

[46] T. Hikihara, H. Ueda, K. Okunishi, K. Harada, T. Nishino, Improving the accuracy of the tree-tensor network approach by optimization of network structure, Phys. Rev. B (2025). (in press), https://journals.aps.org/prb/accepted/10.1103/ljj8-tkpc. arXiv:2501.15514

[47] K. Harada, T. Okubo, N. Kawashima, Tensor tree learns hidden relational structures in data to construct generative models, Mach. Learn. Sci. Technol. 6 (2) (2025) 025002. https://doi.org/10.1088/2632-2153/adc2c7

[48] K.G. Wilson, The renormalization group: critical phenomena and the Kondo problem, Rev. Mod. Phys. 47 (4) (1975) 773–840. https://doi.org/10.1103/RevModPhys.47.773

[49] T. Hikihara, A. Furusaki, M. Sigrist, et al., Numerical renormalization-group study of spin correlations in one-dimensional random spin chains, Phys. Rev. B 60 (17) (1999) 12116–12124. https://doi.org/10.1103/PhysRevB.60.12116

[50] Specifically, Specifically, each sweep searches for the lowest-energy state with M', so that the renormalized wave function after the sweep span on the subspaces labeled by $\{\dots, M'-1, M', M'+1, \dots\}$ as long as $\chi_{\rm init} > 1$. 2025.

[51] R. Watanabe, H. Ueda, Automatic structural search of tensor network states including entanglement renormalization, Phys. Rev. Res. 6 (2024) 033259. https://doi.org/10.1103/PhysRevResearch.6.033259

[52] There should be caution about discrepancies between the isometry indices i in the main text and the output format. When TTNOpt outputs the physical properties on “basic.csv”, indices of isometries $i \in [0, N\_t-1]$ are shifted by adding the system size N to assign physical sites as nodes with index from $[0, N-1]$. 2025.

[53] S.-k. Ma, C. Dasgupta, C.-k. Hu, Random antiferromagnetic chain, Phys. Rev. Lett. 43 (1979) 1434–1437. https://doi.org/10.1103/PhysRevLett.43.1434

[54] J. Tindall, M. Stoudenmire, R. Levy, 2024. arXiv:2410.03572. https://doi.org/10.48550/arXiv.2410.03572

[55] H. Manabe, Y. Sano, 2024. arXiv:2412.12067. https://doi.org/10.48550/arXiv.2412.12067

[56] H. Shinaoka, M. Wallerberger, Y. Murakami, K. Nogaki, R. Sakurai, P. Werner, A. Kauch, Multiscale space-time ansatz for correlation functions of quantum systems based on quantics tensor trains, Phys. Rev. X 13 (2023) 021015. https://doi.org/10.1103/PhysRevX.13.021015

[57] Y.N. Fernández, M.K. Ritter, M. Jeannin, J.-W. Li, T. Kloss, T. Louvet, S. Terasaki, O. Parcollet, J. von Delft, H. Shinaoka, X. Waintal, 2024, arXiv:2407.02454. https://doi.org/10.48550/arXiv.2407.02454

[58] It is worth mentioning that the MPN constructed by the TCI must be transformed into the mixed canonical MPN, since it has the pivots on auxiliary bonds and does not follow the rule of TTNOpt as an input TTN. We put the TCI code in “sample/reconstruct” in TTNOpt such that users can use for broad data., 2025.

[59] N. Nakatani, G.K.-L. Chan, Efficient tree tensor network states (TTNS) for quantum chemistry: generalizations of the density matrix renormalization group algorithm, J. Chem. Phys. 138 (13) (2013) 134113. https://doi.org/10.1063/1.4798639

[60] V. Murg, F. Verstraete, R. Schneider, P.R. Nagy, Ö. Legeza, et al., Tree tensor network state with variable tensor order: an efficient multireference method for strongly correlated systems, J. Chem. Theory Comput. 11 (3) (2015) 1027–1036. https://doi.org/10.1021/ct501187j

[61] J. Haegeman, J.I. Cirac, T.J. Osborne, I. Pižorn, H. Verschelde, F. Verstraete, Time-dependent variational principle for quantum lattices, Phys. Rev. Lett. 107 (2011) 070601. https://doi.org/10.1103/PhysRevLett.107.070601

[62] J. Haegeman, C. Lubich, I. Oseledets, B. Vandereycken, F. Verstraete, Unifying time evolution and optimization with matrix product states, Phys. Rev. B 94 (2016) 165116. https://doi.org/10.1103/PhysRevB.94.165116

[63] D. Bauernfeind, M. Aichhorn, Time dependent variational principle for tree tensor networks, SciPost Phys. 8 (2020) 024. https://doi.org/10.21468/SciPostPhys.8.2.024

[64] S. Sugawara, K. Inomata, T. Okubo, S. Todo, 2025, arXiv:2501.18856. https://doi.org/10.48550/arXiv.2501.18856

[65] J. Schuhmacher, M. Ballarin, A. Baiardi, G. Magnifico, F. Tacchino, S. Montangero, I. Tavernelli, Hybrid tree tensor networks for quantum simulation, PRX Quantum 6 (2025) 010320. https://doi.org/10.1103/PRXQuantum.6.010320