MULTISCALE MODEL. SIMUL.
Vol. 15, No. 4, pp. 1423–1447

TENSOR NETWORK SKELETONIZATION[*]

LEXING YING[†]

Abstract. We introduce a new coarse-graining algorithm, tensor network skeletonization, for the numerical computation of tensor networks. This approach utilizes a structure-preserving skeletonization procedure to remove short-range entanglements effectively at every scale. This approach is first presented in the setting of a two-dimensional (2D) statistical Ising model and is then extended to higher-dimensional tensor networks and disordered systems. When applied to the Euclidean path integral formulation, this approach also gives rise to new efficient representations of the ground states for 1D and 2D quantum Ising models.

Key words. tensor networks, coarse-graining, Ising models, impurity methods, skeletonization

AMS subject classifications. 65Z05, 82B28, 82B80

DOI. 10.1137/16M1082676

1. Introduction.

This paper is concerned with the numerical computation of tensor networks (see [9] for a good introduction of tensor networks). Tensor networks have received a lot of attention in computational statistical mechanics and quantum mechanics as they offer a convenient and effective framework for representing essential quantities.

For example, in classical statistical mechanics, the partition functions are defined as a sum of discrete terms, where the number of terms typically grows exponentially with the number of degrees of freedom. For systems with local Hamiltonians, the tensor networks are able to represent the partition functions with a complexity that grows linearly with the number of the degrees of freedom.

In many quantum many-body systems (especially those with gapped local Hamiltonians), the low energy eigenstates satisfy the area-law, i.e., the entanglement entropy of a part of the system tends to scale linearly with the size of the boundary of the part rather than its volume. These area-law states form an exponentially small part of the high-dimensional Hilbert space. Tensor networks (e.g., density matrix renormalization group [16, 12, 11], matrix product states (MPS) [10], projected entangled pair states [13]), which are designed to target these area-law states, have been used extensively in variational optimizations for these low energy states. In addition, recent work has shown that, through the Euclidean path integral formulations, the ground and thermal states of systems with local Hamiltonians can be well approximated by tensor networks.

1.1. Definition.

A tensor network is associated with a triple $(V, E, \{T^i\}_{i\in V})$. Here, the following hold:

- V is a set of vertices where the degree of the vertex $i \in V$ is denoted as d_i.
- E is a set of edges where edge $e \in E$ is associated with an integer called *bond dimension* χ_e. As we shall see, the bond dimension determines the

[*]Received by the editors July 1, 2016; accepted for publication (in revised form) May 12, 2017; published electronically October 24, 2017.
http://www.siam.org/journals/mms/15-4/M108267.html
Funding: This work was partially supported by the National Science Foundation under award DMS-1521830 and the U.S. Department of Energy Advanced Scientific Computing Research program under award DE-FC02-13ER26134/DE-SC0009409.

[†]Department of Mathematics and Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA 94305 (lexing@stanford.edu).

dimensions of the tensors to be defined at the vertices. Here we allow an edge to be linked with either two or one vertice. An edge linked with two vertices is called *interior*, while an edge with only one vertex is called *boundary.* As a result, the edge set E is the disjoint union $E = E_I \cup E_B$, where E_I is the set of the interior edges and E_B is the set of the boundary edges.

- For each vertex $i \in V$, T^i is a tensor of order d_i, or in short a d_i-tensor, where d_i is the degree of i. Each of the d_i directions of T^i is associated with one of the adjacent edges of vertex i and the dimension of this direction is equal to the bond dimension of the associated edge. For example, if the adjacent edges are denoted by $e_1, e_2, \ldots, e_p$ with $p = d_i$, then T^i is a tensor of size $\chi_{e_1} \times \chi_{e_2} \times \ldots \times \chi_{e_p}$.

Once the triple $(V, E, \{T^i\}_{i \in V})$ is specified, the tensor network represents a tensor that is obtained via contracting all interior edges in E_I. The result is an $|E_B|$-tensor, denoted as

$$\operatorname{tr}_{E_I}\left(\bigotimes_{i \in V} T^i\right). \tag{1}$$

When the set of boundary edges E_B is empty, the tensor network contracts to a scalar. The main goal of this paper is to compute (1) in an accurate and efficient way. Throughout the paper, we follow the following notational conventions:

- The lowercase letters i and j are used to denote vertices in V.
- The lowercase letters a, b, c, d, e, and f are used to denote the edges in E.
- The uppercase letters T, U, and V are used to denote tensors.

The framework of tensor networks is a powerful tool since mathematically it offers efficient representations for high-dimensional functions or probability distributions with certain underlying geometric structures. As an example, let us consider the two-dimensional (2D) statistical Ising model on a periodic square lattice. The vertex set V consists of the lattice points of an $n \times n$ Cartesian grid and $N = n \times n$ is the number of vertices. The edge set E consists of the edges between horizontal and vertical neighbors, defined using the periodic boundary condition (see Figure 1(a)). Here $|E| = 2N$, $E_I = E$, and $E_B = \emptyset$.

At temperature T, the partition function $Z_N(\beta)$ for the inverse temperature $\beta = 1/T$ is given by

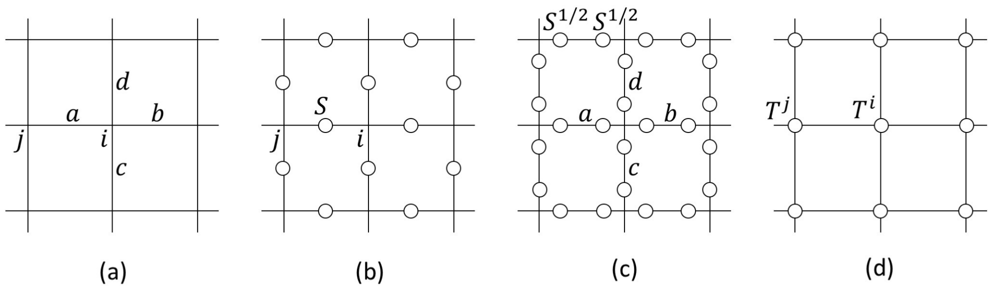

FIG. 1. *Representing the partition function $Z_N(\beta)$ of 2D statistical Ising model using a tensor network.* (a) *The vertices and edges of the tensor network.* (b) *The 2-tensor S associated with each edge.* (c) *Introducing $S^{1/2}$ splits S into the product of two 2-tensors.* (d) *Contracting the four 2-tensors adjacent to a vertex i forms the 4-tensor T^i.*

$$Z_N(\beta) = \sum_{\sigma} e^{-\beta H_N(\sigma)}, \quad H_N(\sigma) = -\sum_{(ij)\in E} \sigma_i \sigma_j,$$

where $\sigma = (\sigma_1, \dots, \sigma_N)$ stands for a spin configuration at N vertices with $\sigma_i = \pm 1$ and the sum of σ is taken over all 2^N configurations. Here (ij) denotes the edge between two adjacent vertices i and j and the sum in $H_N(\sigma)$ is over these $2N$ edges.

In order to write $Z_N(\beta)$ in the form of a tensor network, one approach is to introduce a 2×2 matrix S

$$S = \begin{pmatrix} e^{\beta} & e^{-\beta} \\ e^{-\beta} & e^{\beta} \end{pmatrix},$$

which is the multiplicative term in $Z_N(\beta)$ associated with an edge between two adjacent vertices. The partition function $Z_N(\beta)$ is built from the S matrices over all edges in E (see Figure 1(b)). Since S is a symmetric matrix, its symmetric square root $S^{1/2}$ is well defined with the following elementwise identity:

$$S_{ij} = \sum_{a} S^{1/2}_{ia} S^{1/2}_{aj},$$

where a denotes the edge that connects i and j (see Figure 1(c)). Here and throughout the paper, the lowercase letters (e.g., i, j, and k) for denoting a vertex in V are also used for the running index associated with that vertex. The same applies to the edges: the lowercase letters (e.g., e and f) for denoting an edge in E are also used as the running index associated with that edge.

At each vertex i, one can then introduce a 4-tensor T^i

$$T^i_{abcd} = \sum_{i} S^{1/2}_{ia} S^{1/2}_{ib} S^{1/2}_{ic} S^{1/2}_{id}, \tag{2}$$

which essentially contracts the four $S^{1/2}$ tensors adjacent to the vertex i (see Figure 1(c)). Finally, the partition function $Z_N(\beta)$ can be written as

$$Z_N(\beta) = \mathrm{tr}_E \left(\bigotimes_{i \in V} T^i \right)$$

(see Figure 1(d)).

1.2. Previous work. One of the main computational tasks is to evaluate tensor networks (1) accurately and efficiently. When the boundary edge set E_B is empty, this amounts to computing the scalar given by (1). In the general case, this means approximating an initial tensor network with a more compact one (typically with a small number of vertices and edges as well as with smaller bond dimensions). Naive tensor contractions following the definition (1) are computationally prohibitive as the bond dimensions can scale exponentially with the number of vertices. Therefore, the key is to keep the bond dimensions low while maintaining good approximation accuracy.

In recent years, there has been a lot of work devoted to efficient algorithms for evaluating tensor networks. In [7], Levin and Nave introduced the tensor renormalization group (TRG) as probably the first practical algorithm for this task. When applied to the 2D statistical Ising models, this method utilizes an alternating sequence of tensor contractions and singular value decompositions. However, one problem with TRG is the accumulation of so-called short-range entanglement, which increases the bond dimensions and computational costs dramatically as the method proceeds. The concept of short-range entanglement is not precisely defined in the physics literature and for the purpose of this paper we talk about short-range entanglements associated with a local part of the tensor network. The existence of such short-range entanglement refers to the situation that *the tensors in a local tensor network can be replaced with simplified tensors (with smaller dimensions) without affecting significantly the numerical values of the tensor represented by the local tensor network*.

In a series of papers [18, 17, 21], Xiang and others introduced the higher order tensor renormalization group (HOTRG) as an extension of TRG to address 3D classical spin systems. The same group has also introduced the second renormalization group (SRG) based on the idea of approximating the environment of a local tensor before performing reductions. Though SRG typically gives more accurate results, the computation time of SRG tends to grow significantly with the size of the local environment.

In [5], Gu and Wen introduced the method of tensor entanglement filtering renormalization (TEFR) as an improvement of TRG for 2D systems. Compared with TRG, this method makes an extra effort in removing short-range entanglements and hence produces more accurate and efficient computation.

Recently in [3, 4], Evenbly and Vidal proposed the tensor network renormalization (TNR). The key step of TNR is to apply the so-called disentanglers to remove short-range entanglement. These disentanglers appeared earlier in the work of the multiscale entanglement renormalization ansatz (MERA) [15]. For a fixed bond dimension, TNR gives significantly more accurate results compared to TRG at the cost of increasing the computational complexity. However, at this point it is not clear how to extend the approach of TNR to systems in higher dimensions.

More recently in [20], Yang, Gu, and Wen proposed the loop tensor network renormalization (Loop-TNR), which improves on TEFR by removing short-range entanglement more efficiently using well-developed variational MPS routines. While producing results similar to the ones of TNR in [3], this algorithm is more efficient, partly due to the leverage of these efficient MPS algorithms.

1.3. Contribution and outline. These recent developments have significantly improved the efficiency and accuracy of the computation of tensor networks. From a computational point of view, it would be great to have a general framework that allow for extensions to

- tensor networks of 3D and 4D systems and
- systems that are not translationally invariant, such as disordered systems.

Building on top of the previous work in the physics literature, we introduce a new coarse-graining approach, called the tensor network skeletonization (TNSK), as a first step toward building such a general framework. At the heart of this approach is a new procedure called the *structure-preserving skeletonization*, which removes short-range entanglement efficiently while maintaining the structure of a local tensor network. This allows us to generalize TNSK quite straightforwardly to spin systems of higher dimensions. In addition, we also provide a simple and efficient algorithm for performing the structure-preserving skeletonization. This allows for applying TNSK efficiently to systems that are not translationally invariant.

The rest of this paper is organized as follows. Section 2 summarizes the basic tools used by the usual tensor network algorithms. Section 3 is the main part of the paper and explains TNSK for 2D statistical Ising model. Section 4 extends the algorithm to the 3D statistical Ising model. Section 5 discusses how to build efficient representations of the ground states of 1D and 2D quantum Ising models using TNSK. Finally, section 6 discusses some future work.

2. Basic tools.

2.1. Local replacement.

The basic building blocks of all tensor network algorithms are *local replacements*. Suppose that vertex V and edge set E of a tensor network $(V, E, \{T^i\}_{i\in V})$ are partitioned as follows:

$$V = V_1 \cup V_2, \quad E = E_1 \cup E_2 \cup E_{12},$$

where E_1 and E_2 are the sets of interior edges of V_1 and V_2, respectively, and E_{12} is the set of edges that link across V_1 and V_2. Such a partition immediately gives an identity

$$\mathrm{tr}_E \left(\bigotimes_{i\in V} T^i\right) = \mathrm{tr}_{E_2\cup E_{12}} \left(\left(\bigotimes_{i\in V_2} T^i\right) \bigotimes \mathrm{tr}_{E_1} \left(\bigotimes_{i\in V_1} T^i\right)\right). \tag{3}$$

Assume now that there exists another tensor network B for which the following approximation holds:

$$B \approx \mathrm{tr}_{E_1} \left(\bigotimes_{i\in V_1} T^i\right)$$

(see Figure 2(a)). Typically B is much simpler in terms of the number of the vertices and/or the bond dimensions of the edges. In most situations, the above approximation refers to the Frobenius norm.

A *local replacement* refers to replacing $\mathrm{tr}_{E_1}\left(\bigotimes_{i\in V_1} T^i\right)$ in (3) with B to get a simplified approximation

$$\mathrm{tr}_E \left(\bigotimes_{i\in V} T^i\right) \approx \mathrm{tr}_{E_2\cup E_{12}} \left(\left(\bigotimes_{i\in V_2} T^i\right) \bigotimes B\right)$$

(see Figure 2(b)). Most algorithms for tensor networks apply different types of local replacements successively until the tensor network is simplified to a scalar or left with only boundary edges.

The simplest instance of local replacement is the *tensor contraction* and it simply combines two adjacent tensors into a single one. As an example, let P be a 2-tensor

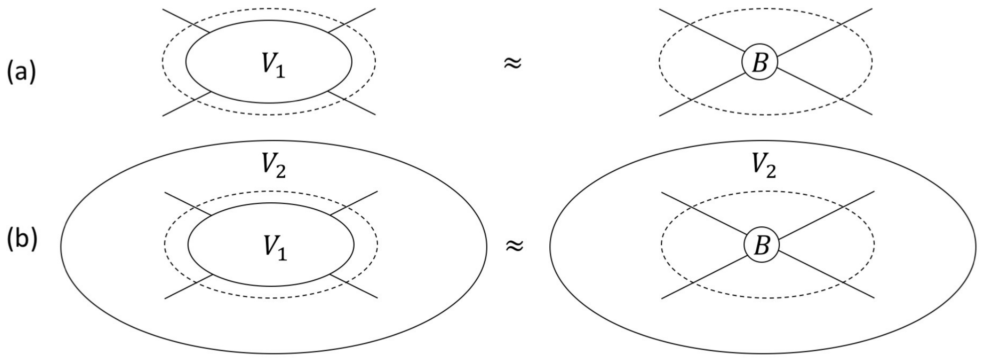

FIG. 2. *Local replacement.* (a) *Part of the tensor network associated with vertices in* V_1 *is approximated by a simplified tensor network* B. (b) *Locally replacing* V_1 *with* B *results in a approximation of the whole tensor network.*

adjacent to edges a and c and Q be another 2-tensor adjacent to edges c and b (see Figure 3(a)). The resulting 2-tensor T obtained from contracting P and Q is simply the product of P and Q, i.e.,

$$T_{ab} = \sum_c P_{ac} Q_{cb}$$

(see Figure 3(a)). Often when the contraction is applied, the edges a and b often come from grouping a set of multiple edges.

A second instance is called the *projection*. Typically it is carried out by performing a singular value decomposition of T followed by thresholding small singular values, i.e.,

$$T \approx USV^{\mathsf{T}}, \quad T_{ab} \approx \sum_{cd} U_{ac} S_{cd} V_{bd}$$

(see Figure 3(b)). Here U and V are both orthogonal matrices and S is a diagonal matrix. Due to the truncation of small singular values, the bond dimensions at edges c and d can be significantly smaller compared to those of a and b. Throughout this paper, each orthogonal matrix shall be denoted by a diamond in the figures. As with the contraction, each of the indices a and b often comes from grouping a set of multiple edges. The SVD-based projection can also be modified slightly to give a few equivalent forms (see Figure 3(b))

$$\begin{aligned} T &\approx UU^{\mathsf{T}}T, \quad T_{ab} \approx \sum_{ce} U_{ac} U_{ec} T_{eb}, \\ T &\approx UR, \quad R = U^{\mathsf{T}}T, \quad T_{ab} \approx \sum_c U_{ac} R_{cb}. \end{aligned}$$

In the rest of this paper, we refer to the first one as the $UU^{\mathsf{T}}T$-projection and the second one as the UR-projection. In Figure 3 and the following figures, we choose to omit the transpose sign since it can be inferred from the bond dimensions of the adjacent edges. In addition, the transpose fails to make sense when we start working with tensors with order greater than two.

Another instance of local replacements uses the *disentanglers* introduced in [15], which are specially designed isometries (unitary operators) obtained through optimization in order to isolate or remove short-range entanglements. These disentanglers play a key role in the work of TNR [3] as mentioned above. Since the TNSK

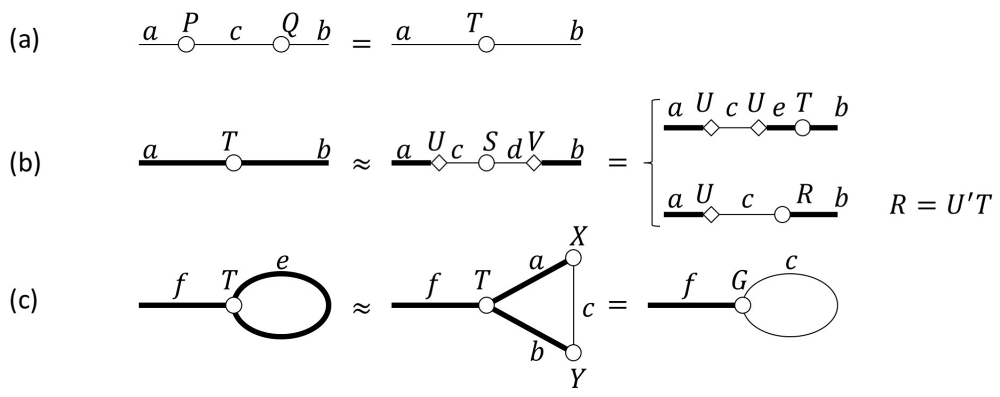

FIG. 3. *Instances of local replacements.* (a) *Contraction.* (b) *Projection.* (c) *Loop skeletonization.*

approach of this paper does not depend on the disentanglers, we refer to [3, 14, 2] for more detailed discussions of them.

2.2. Loop skeletonization. At the center of the TNSK approach is a new type of local replacement called the *loop skeletonization*. The overall goal is to reduce the bond dimensions of the interior edges of a loopy local tensor network without changing its topology. Let us consider a 3-tensor T with two of its directions marked with a same edge e (and thus to be contracted). The loop skeletonization seeks two 2-tensors X and Y (see Figure 3(c)) such that

$$\operatorname{tr}_e T \approx \operatorname{tr}_{abc}(X \otimes T \otimes Y), \quad \sum_e T_{eef} \approx \sum_{abc} X_{ac} T_{abf} Y_{bc}, \tag{4}$$

where the bond dimension χ_c of edge c should be significantly smaller compared to χ_e of edge e. This is possible because there might exist short-range entanglements within the loop that can be removed from the viewpoint of the exterior of this local tensor network.

A convenient reformulation of the problem is to view T as a $\chi_e \times \chi_e$ matrix where each entry T_{ab} is a χ_f-dimensional vector and view X and Y as matrices. Then one can rewrite the condition in (4) as

$$\operatorname{tr}_e T \approx \operatorname{tr}_c(X^\mathsf{T} T Y), \tag{5}$$

where the products between X, T, and Y are understood as matrix multiplications. The approximation accuracy depends on the bond dimension χ_c where a larger value of χ_c implies smaller approximation error.

As far as we know, there does not exist a simple and robust numerical linear algebra routine that solves this approximation problem directly. Instead, we propose to solve the following regularized optimization problem:

$$\min_{X,Y} \| \operatorname{tr}_e T - \operatorname{tr}_c(X^\mathsf{T} T Y)\|_2^2 + \alpha(\|X\|_F^2 + \|Y\|_F^2), \tag{6}$$

where α is the regularization parameter. The value of α is chosen to balance the magnitude of X and Y with the approximation error. In the numerical tests in this paper, α is set to be order of 10^{-10}.

Though this optimization problem is nonconvex, it can be solved effectively in practice using the alternating least square algorithm once a good initial guess is available. More precisely, given a initial guess for $X^{(0)}$ and $Y^{(0)}$, one alternates the following two steps for $n = 0, 1, \ldots$ until convergence:

$$\begin{aligned} X^{(n+1)} &= \operatorname{argmin}_X \| \operatorname{tr}_e T - \operatorname{tr}_c(X^\mathsf{T} T Y^{(n)})\|_2^2 + \alpha \|X\|_F^2, \\ Y^{(n+1)} &= \operatorname{argmin}_Y \| \operatorname{tr}_e T - \operatorname{tr}_c((X^{(n+1)})^\mathsf{T} T Y)\|_2^2 + \alpha \|Y\|_F^2. \end{aligned}$$

Since each of the two steps is a least square problem in X or Y, they can be solved efficiently with standard numerical linear algebra routines. Being a nonconvex approach, the performance of the algorithm depends significantly on the initial guess. The following approach for choosing the initial guess works well when the tensor T is symmetric or near symmetric in the directions associated with edge a and b:

- First, reshape T to be a matrix of size $\chi_f \times \chi_e^2$ (notice $\chi_a = \chi_b = \chi_e$). Compute its singular value decomposition and extract the right top singular vector (of length χ_e^2).

- Second, reshape the top singular vector to a $\chi_e \times \chi_e$ matrix. Compute now its singular value decomposition.
- Finally, set both X and Y to be the $\chi_e \times \chi_c$ matrix that consists of the first χ_c left singular vectors.

Our numerical experience shows that, when starting from this initial guess, the alternating least square algorithm converges after a small number of iterations to nearly optimal solutions.

The most computationally intensive steps are the assembly and solution of the least square problems within each iteration. The assembly of the least square problem takes $O\left(\chi_f(\chi_e\chi_c)^2\right)$ steps, while the solution costs $O\left((\chi_e\chi_c)^3\right)$ steps. Therefore, the overall cost is of order $O\left(\chi_f(\chi_e\chi_c)^2 + (\chi_e\chi_c)^3\right)$.

Other types of regularization or convex relaxation can also be used instead. In this paper, we choose to use the Frobenius norm regularization as in (6) due to its simplicity and efficiency (under well-chosen initial conditions).

3. TNSK for 2D statistical Ising models.

We start with a 2D statistical Ising model on an $n \times n$ lattice with the periodic boundary condition. Following the discussion in section 1, we set the vertex set V_0 to be an $n \times n$ Cartesian grid. Each vertex i is identified with a tuple (i_1, i_2) with $i_1, i_2 \in [n] = \{0, 1, \ldots, n-1\}$. The edge set E_0 consists of the edges between horizontal and vertical neighbors of the Cartesian grid modulus periodicity. This setup also gives rise to an $n \times n$ array of plaquettes, each consisting of four vertices. For a plaquette with vertex $i = (i_1, i_2)$ at its lower-left corner, we shall index this plaquette with $i = (i_1, i_2)$ as well. Here $N = n^2$ is the total number of spins and we assume without loss of generality that $n = 2^L$. The four tensors at the corner of each plaquette naturally form a local tensor network, which will play a key role in the algorithms described below.

3.1. Partition function.

Following the discussion in section 1, the partition function $Z_N(\beta)$ can be represented using a tensor network $(V^0, E^0, \{T^i\}_{i\in V_0})$, where T^i are given in (2). Let χ be a predetermined upper bound for the bond dimension of the edges of the tensor network. One can assume without loss of generality that the bond dimension χ_e for the edge $e \in E^0$ is close to this constant χ. When χ is significantly larger than 2, this can be achieved by contracting each 2×2 neighborhood of tensors into a single tensor. For example, when $\chi = 4$, one round of such contractions brings $\chi_e = \chi = 4$.

3.1.1. Algorithm.

The TNSK algorithm consists of a sequence of coarse-graining iterations. At the beginning of the ℓth iteration, one holds a tensor network $(V^\ell, E^\ell, \{T^i\}_{i\in V_\ell})$ at level ℓ with $(n/2^\ell) \times (n/2^\ell)$ vertices.

With the exception of the 0th iteration, we require the following iteration invariance to hold at the beginning of each iteration:

- For each plaquette with index equal to $(0,0)$ or $(1,1)$ modulus 2, the short-range entanglement within this plaquette has already been eliminated (see Figure 4(a)).

Here the short-range entanglement within a plaquette refers to the part of its local tensor network that can be eliminated without any boundary edge. Therefore from the viewpoint of the boundary edges of the plaquette's local tensor network, this short-range entanglement is invisible. Thus it can and should be eliminated in order to keep bond dimensions small.

In what follows, we refer to the plaquettes with index equal to $(0,0)$ modulus 2 as $(0,0)_2$ plaquettes and similarly those with index equal to $(1,1)$ modulus 2 as

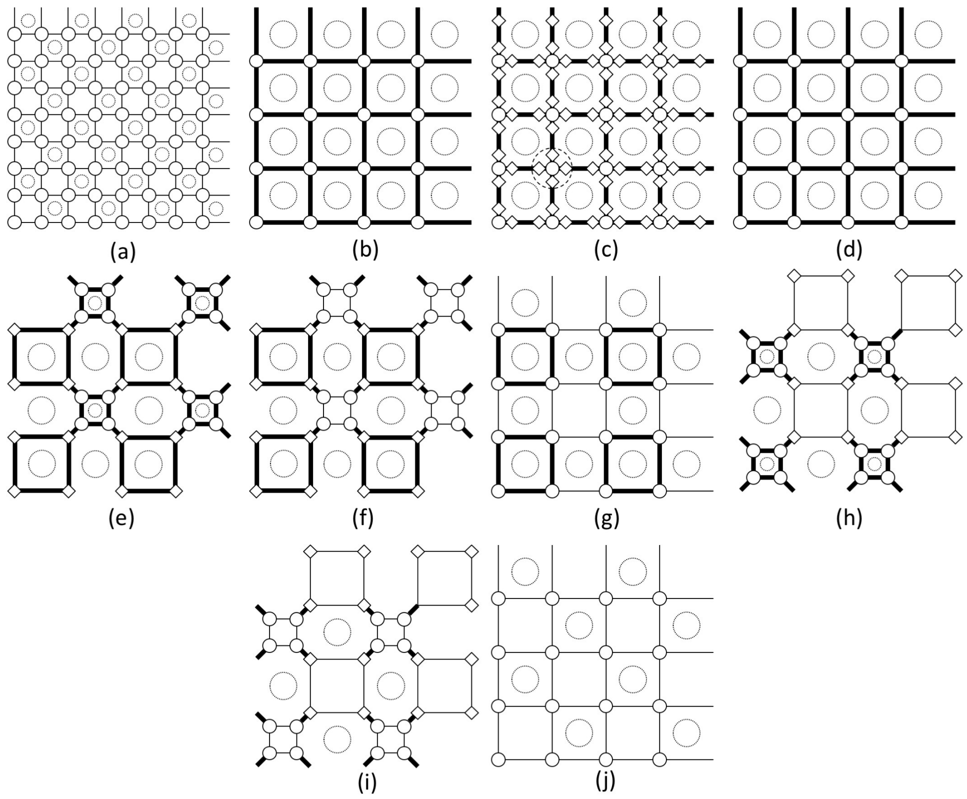

Fig. 4. *A single iteration of the tensor network skeletonization algorithm. The starting point is a tensor network with bond dimension χ and short-range entanglement removed in the $(0,0)_2$ and $(1,1)_2$ plaquettes. The final result is a coarse-grained tensor network with 1/4 of the vertices (or tensors). This coarse-grained tensor network also has bond dimensions equal to χ with short-range entanglement removed in the (larger) $(0,0)_2$ and $(1,1)_2$ plaquettes. The bold lines stand for edges with bond dimensions equal to $O(\chi^2)$.*

$(1,1)_2$ plaquettes. In Figure 4(a), the dotted circles denote the existence of short-range entanglement. Notice that these circles do not appear in the $(0,0)_2$ plaquettes and $(1,1)_2$ plaquettes.

The ℓth iteration consists of the following steps:

1. Merge the tensors at the four vertices of each $(0,0)_2$ plaquette into a single tensor (see Figure 4(b)). This requires a couple of contractions defined in section 2. The $(1,1)_2$ plaquettes are stretched and this results in a new graph that contains only 1/4 of the vertices. The tensors at the new vertices are identical but the bond dimension of the new edges is equal to χ^2 (shown using the bold lines). Since these $(1,1)_2$ plaquettes at level ℓ do not contain short-range entanglement at level ℓ, no short-range entanglement of level ℓ will survive at level $\ell+1$. However, there are new short-range entanglement of level $\ell+1$ in the tensor network and these are marked with larger dotted circles inside the new plaquettes at level $\ell+1$. The key task for the rest of the iteration is to remove part of these short-range entanglements at level $\ell+1$ and reduce the bond dimension from χ^2 back to χ at the same time.
2. For each vertex i in Figure 4(b), denote the tensor at i by T^i_{abcd}, where a, b, c, and d refer to the left, right, bottom, and top edges. Apply two $UU^\mathsf{T}T$-

projections to T^i_{abcd}. Here the first projection is with the left edge versus the rest edges and the second one is with the bottom edge versus the rest. These two projections effectively insert two orthogonal (diamond) matrices in each edge (see Figure 4(c)). Most TRG algorithms utilize this step to reduce the bond dimension directly from χ^2 back to χ, thus incurring a significant error. Here, however, the bond dimension after the projection is kept close to χ^2. This introduces a much smaller truncation error as compared to the TRG algorithms.

3. At each vertex i in Figure 4(c), merge the tensor T^i_{abcd} with its four adjacent orthogonal (diamond) matrices (see Figure 4(d)). Though the tensor network obtained after this step has the same topology as the one in Figure 4(b), the bond dimension is somewhat reduced (less than χ^2 but still on the same order). Another key point is that these two recent steps effectively provide a unitary gauge transformation for the degrees of freedoms associated with the edges. Such a unitary transformation in a certain sense diagonalizes the interaction between the two tensors sharing an edge, thus making it relatively easy to choose a good initial condition for the structure-preserving skeletonization described in section 3.1.2.
4. For each $(1,1)_2$ plaquette in Figure 4(d), apply the UR-projection to the 4-tensor at each of its corners. Here the two edges adjacent to the plaquette are grouped together. Notice that the (round) R tensors are placed close to the $(1,1)_2$ plaquette, while the (diamond) U tensors are placed away from it. Though this projection step does not reduce bond dimensions, it allows us to isolate each $(1,1)_2$ plaquette. The resulting graph is given in Figure 4(e).
5. In this key step, remove the short-range entanglement within each $(1,1)_2$ plaquette. The details of this procedure will be provided below in section 3.1.2. The resulting $(1,1)_2$ plaquette has its short-range entanglement removed and the bond dimensions of its four surrounding edges are reduced from χ^2 back to χ (see Figure 4(f)).
6. For each $(1,1)_2$ plaquette, contract back the UR-projections at each of its four corners. Notice that, due to the previous step, the new R tensors have bond dimensions equal to χ. The resulting tensor network (see Figure 4(g)) is similar to the one in Figure 4(d) but now the short-range entanglements in the $(1,1)_2$ plaquettes are all removed.
7. Now repeat the previous three steps to the $(0,0)_2$ plaquettes. This is illustrated in Figure 4(h), (i), and (j). The resulting tensor network now has short-range entanglement removed in both $(0,0)_2$ and $(1,1)_2$ plaquettes. In addition, the bond dimension of the edges is reduced back to χ from χ^2.

This finishes the ℓth iteration. At this point, one obtains a new tensor network denoted by $(V^{\ell+1}, E^{\ell+1}, \{T^i\}_{i\in V_{\ell+1}})$ that is a self-similar and coarse-grained version of $(V^\ell, E^\ell, \{T^i\}_{i\in V_\ell})$. Since the short-range entanglements in both $(0,0)_2$ and $(1,1)_2$ plaquettes are removed, this new tensor network satisfies the iteration invariance and can serve as the starting point of the $(\ell+1)$th iteration.

Following this process, the TNSK algorithm constructs a sequence of tensor networks

$$(V^\ell, E^\ell, \{T^i\}_{i\in V_\ell}), \quad \ell = 0, 1, 2, \ldots, L.$$

The last one is a single 4-tensor with the left and right edges identified and similarly with the bottom and top edges identified. Contracting this final tensor gives a scalar value for the partition function $Z_N(\beta)$.

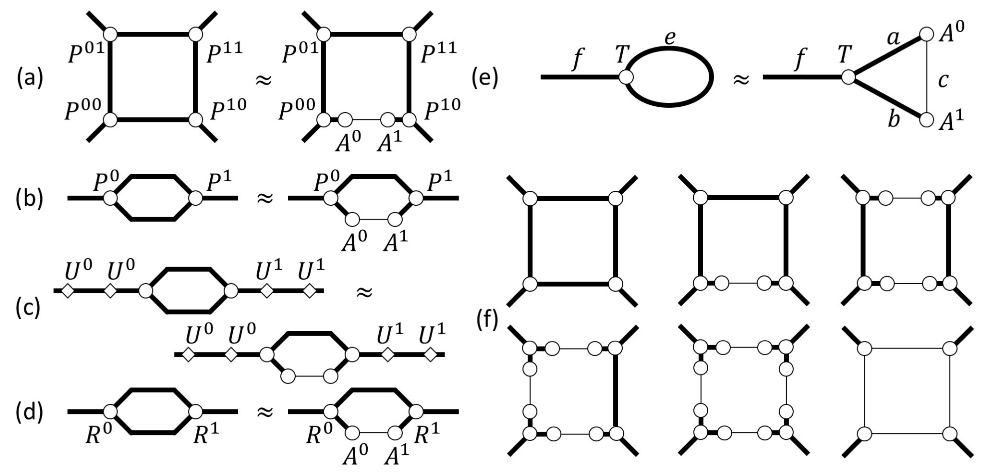

FIG. 5. *The structure-preserving skeletonization procedure removes the short-range entanglement within a* $(1,1)_2$ *(or* $(0,0)_2$*) plaquette. The bold lines stand for edges with bond dimensions* $\geq \chi^2$.

3.1.2. Short-range entanglement removal via structure-preserving skeletonization. In the description of the algorithm in section 3.1.1, the missing piece is how to remove the short-range entanglement of a $(1,1)_2$ plaquette and reduce the bond dimension of its four surrounding edges (from Figure 4(e) to Figure 4(f)).

This is carried out by the *structure-preserving skeletonization* illustrated in Figure 5 with the four corner tensors denoted by P^{00}, P^{10}, P^{01}, and P^{11}. Instead of replacing the four corner 3-tensors simultaneously, this procedure considers the four interior edges one by one and inserts two tensors of size $\chi^2 \times \chi$ in each edge.

Let us start with the bottom edge. Here we seek two 2-tensors A^0 and A^1 of size $\chi^2 \times \chi$ such that the 4-tensor represented by the new $(1,1)_2$-plaquette (after inserting A^0 and A^1) approximates the 4-tensor represented by the original plaquette (see Figure 5(a)).

1. Merge the two left tensors P^{00} and P^{01} into a 3-tensor P^0 and merge the two right tensors P^{10} and P^{11} into a 3-tensor P^1. Notice that the two boundary edges have bond dimension equal to χ^4. The criteria for choosing A^0 and A^1 is now given in Figure 5(b).
2. Since the bond dimensions of the two edges between P^0 and P^1 will eventually be reduced to χ, this implies that the bond dimensions of the two boundary edges can be cut down to $\chi^2 = \chi \times \chi$ without affecting the accuracy. For this, we perform the $UU^{\mathsf{T}}T$-projection to both P^0 and P^1. The criteria for choosing A^0 and A^1 is now given in Figure 5(c).
3. Remove the two tensors U^0 and U^1 at the two endpoints. Merge U^0 with P^0 to obtain a 3-tensor R^0 and similarly merge U^1 with P^1 to obtain a 3-tensor R^1. The criteria for choosing A^0 and A^1 is in Figure 5(d).
4. Finally, contracting the top edge between R^0 and R^1 results in a new 3-tensor T. The criteria for choosing A^0 and A^1 is now given in Figure 5(e). This is exactly the setting of the loop skeletonization in section 2.2 and can be solved efficiently using the alternating least square algorithm proposed there. For the initial guess of A^0 and A^1, we choose them to be the matrices with one on the diagonal and zero everywhere else because the second and third

steps of the algorithm in section 3.1.1 "diagonalize" the interaction between nearby tensors.

At this point, two tensors A^0 and A^1 are successfully inserted into the bottom edge. One can repeat this process now for the top, left, and right edges in a sequential order. Once this is done, merging each of the corner tensors with its two adjacent inserted tensors gives the desired approximation (see Figure 5(f) for this whole process).

A task of reducing the bond dimensions of the four surrounding edges of a plaquette has appeared before in the work of TEFR [5]. However, the algorithms proposed there are different from the one described here and the resulting plaquette was used in a TRG setting that does not extend naturally to 3D tensor network problems.

3.1.3. Complexity estimate. In each iteration of the above algorithm, the most computationally intensive parts are step 4 of the algorithm in section 3.1.1 and step 2 of the structure-preserving skeletonization in section 3.1.2. In both steps, we perform a singular value decomposition of size $\chi^4 \times \chi^4$ and the computation cost scales like $O(\chi^{12})$. Another key step is the alternating least square algorithm used in the loop skeletonization. In the current setting, the parameters used in the loop skeletonization are given by $\chi_f = \chi^4$, $\chi_e = \chi^2$, and $\chi_c = \chi$. Therefore, the overall cost of the alternating least square algorithm scales like $O(n_i \cdot \chi^{10})$, where n_i is the number of alternating iterations.

As the algorithm in section 3.1.1 for computing the partition function takes $L = \log N$ iterations, where N is the number of vertices, the overall complexity of the algorithm scales like $O\left((\chi^{12} + n_i \cdot \chi^{10}) \cdot \log N\right)$.

We would like to remark that the physical meaning of the bond dimension χ in TNSK is different from most of the TRG algorithms as we do not perform the 45 degree rotations. Numerical results show that, for a similar accuracy, the bond dimension required by this algorithm is roughly the square root of the one used in most TRG algorithms. For example, this algorithm with $\chi = 4$ achieves better accuracy than most TRG algorithms with $\chi = 16$ (see the numerical results in section 3.1.5). Therefore, the exponent of the complexity estimate for TNSK should be halved when compared with the complexity analysis of most TRG algorithms.

3.1.4. A modified version. In terms of coarse-graining the tensor network, each iteration of the algorithm in section 3.1.1 achieves the important goal of constructing a self-similar version while keeping the bond dimension constant (equal to χ) (see Figure 4(a) and (j) for comparison).

However, for the purpose of merely computing the partition function $Z_N(\beta)$, one part of the work is redundant. More specifically, at the end of the ℓth iteration, the structure-preserving skeletonization is also performed for the $(0,0)_2$-plaquettes at level $\ell + 1$ to remove their short-range entanglements. However, right at the beginning of the next iteration, a merging step contracts the four corner tensors of each $(0,0)_2$-plaquette. By eliminating this structure-preserving skeletonization for the $(0,0)_2$ plaquettes, one obtains a modified version of the algorithm (see Figure 6) that can potentially be computationally more efficient.

Compared with the algorithm illustrated in Figure 4, the main differences are as follows:

- The iteration invariance is that, at the beginning of each iteration, only the short-range entanglements of the $(1,1)_2$ plaquettes are removed. Therefore, the bond dimensions of the edges around a $(0,0)_2$ plaquette are equal to χ^2. This is not so appealing from the viewpoint of approximating a tensor network

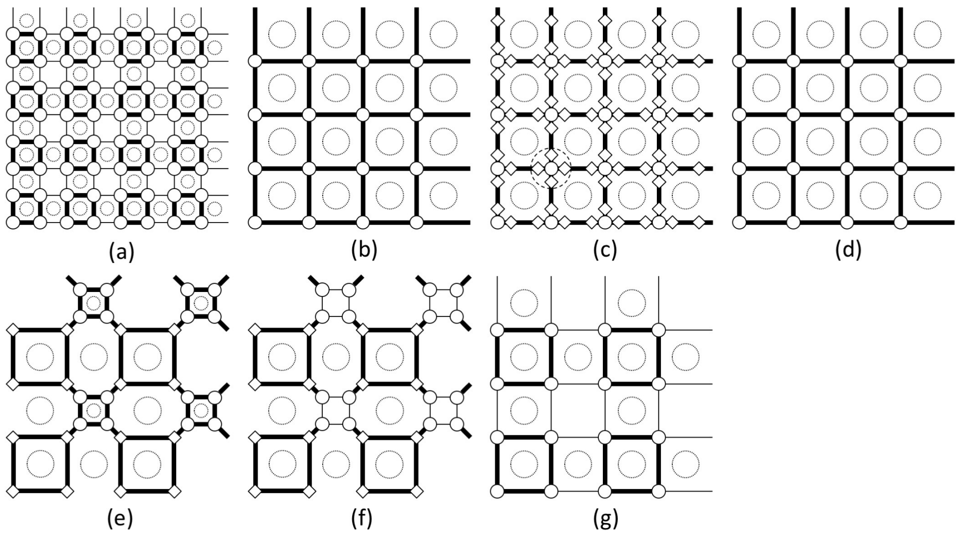

Fig. 6. *A single iteration of the modified tensor network skeletonization algorithm. The starting point is a tensor network with short-range entanglement removed in* $(1,1)_2$ *plaquettes. The final point is a coarse-grained tensor network with* $1/4$ *vertices (tensors). This coarse-grained tensor also has bond dimensions equal to* χ *around the* $(1,1)_2$ *plaquettes and* χ^2 *around the* $(0,0)_2$ *plaquettes. The short-range entanglement is removed for the (larger)* $(1,1)_2$ *plaquettes. The bold lines stand for edges with bond dimensions* χ^2.

with minimal bond dimension. However, as one can see from Figure 6(a) and (b), a contraction step is applied immediately to these $(0,0)_2$ plaquettes so that the high bond dimensions do not affect subsequent computations.

- In Figure 6(e) and (f), the structure-preserving skeletonization is applied only to the $(1,1)_2$ plaquettes.
- In Figure 6(g), the resulting tensor network at level $\ell+1$ satisfies the new iteration invariance and hence it can serve as the starting point of the next iteration.

As we shall see in section 3.2.1, this modified algorithm also has the benefit of incurring minimum modification when evaluating observables using the impurity method.

3.1.5. Numerical results. Let us denote by $\tilde{Z}_N(\beta)$ the numerical approximation of the partition function $Z_N(\beta)$ obtained via TNSK. The exact free energy per site $f_N(\beta)$ and the approximate free energy per site $\tilde{f}_N(\beta)$ are defined by

$$f_N(\beta) = \left(-\frac{1}{\beta}\log Z_N(\beta)\right)/N, \quad \tilde{f}_N(\beta) = \left(-\frac{1}{\beta}\log \tilde{Z}_N(\beta)\right)/N.$$

For an infinite 2D statistical Ising system, the free energy per site

$$f(\beta) = \lim_{N\to\infty} f_N(\beta)$$

can be derived analytically [6]. Therefore, for sufficiently large N, $f_N(\beta)$ is well approximated by $f(\beta)$. In order to measure the accuracy of TNSK for computing the partition function, we define the relative error

$$\delta f_N(\beta) \equiv \frac{|\tilde{f}_N(\beta) - f(\beta)|}{|f(\beta)|} \approx \frac{|\tilde{f}_N(\beta) - f_N(\beta)|}{|f_N(\beta)|}.$$

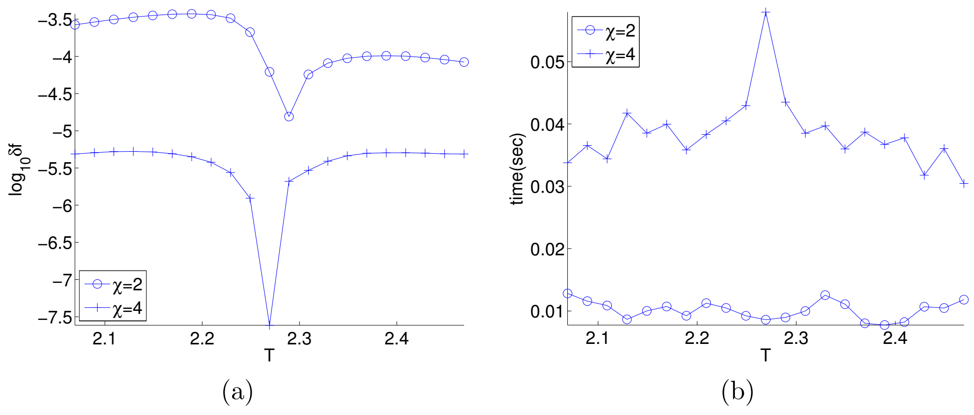

FIG. 7. *Results of free energy calculation.* (a) *The relative error $\delta f_N(\beta)$ of the free energy per site at temperatures around T_c for $\chi = 2, 4$.* (b) *The running time per iteration of the TNSK algorithm in section* 3.1.1 *for the same T and χ values.*

The critical temperature of the 2D statistical Ising model is $T_c = 2/\ln(1+\sqrt{2})$. For a periodic statistical Ising model on a $2^{15} \times 2^{15}$ lattice, Figure 7 plots the relative error (left) and the running time per iteration of TNSK (right) for $\chi = 2, 4$ at different temperatures near the critical temperature T_c.

From the plots in Figure 7 one can make the following observations:

- First, TNSK removes the short-range entanglement quite effectively. With $\chi = 4$, it achieves five or six digits of accuracy for the relative free energy per site. Even with $\chi = 2$, one obtains three or four digits of accuracy.
- Second, TNSK is quite efficiently. For $\chi = 4$, each iteration of the TNSK takes about 0.05 seconds. The running time tends to grow a bit when T approaches the critical temperature T_c. The explanation is that when T is away from the critical temperature, the effective bond dimensions of the tensors reduce to either 1 (above the critical temperature) or 2 (below the critical temperature). Near the critical temperature, the bond dimensions will remain as χ throughout the iterations. This is the main reason that the computation time is higher near T_c. In addition, at criticality, the alternating least square algorithms also take a few more iterations to converge.
- Most surprisingly, for a fixed χ value, TNSK gives more accurate results when the temperature is close to T_c. For example, with $\chi = 4$ at $T = T_c$, the relative error is on the order of 10^{-8}. This is quite different from most of the TRG algorithms where the accuracy deteriorates significantly near T_c.

3.2. Observables.

The TNSK algorithm described in section 3.1 for computing the partition function (and equivalently the free energy) can be extended to compute observables such as the average magnetization and the internal energy per site.

The internal energy $U_N(\beta)$ of the whole system and the internal energy per site $u_N(\beta)$ are defined as

$$U_N(\beta) = \partial_\beta(-\log Z_N(\beta)) = -\frac{\partial_\beta Z_N(\beta)}{Z_N(\beta)}, \quad u_N(\beta) = \frac{U_N(\beta)}{N}.$$

A direct calculation shows that

$$\partial_\beta Z_N(\beta)=\sum_\sigma e^{-\beta H_N(\sigma)}(-H_N(\sigma))=\sum_\sigma \left(\sum_{(ij)}\sigma_i\sigma_j\right) e^{-\beta H_N(\sigma)}=N_e\sum_\sigma(\sigma_i\sigma_j)e^{-\beta H_N(\sigma)},$$

where in the last formula (i,j) can be any edge due to the translational invariance of the system and $N_e=2N$. This gives the following formula for the internal energy per site:

$$u_N(\beta)=\frac{U_N(\beta)}{N}=\frac{N_e}{N}\cdot\frac{\sum_\sigma(\sigma_i\sigma_j)e^{-\beta H_N(\sigma)}}{\sum_\sigma e^{-\beta H_N(\sigma)}}=2\frac{\sum_\sigma(\sigma_i\sigma_j)e^{-\beta H_N(\sigma)}}{\sum_\sigma e^{-\beta H_N(\sigma)}}. \tag{7}$$

To define the average magnetization, one introduces a small external magnetic field B and defines the partition function of the perturbed system

$$Z_{N,B}(\beta)=\sum_\sigma e^{-\beta H_{N,B}(\sigma)},\quad H_{N,B}(\sigma)=-\left(\sum_{(ij)}\sigma_i\sigma_j+B\sum_i\sigma_i\right).$$

The magnetization at a single site i is equal to

$$\langle\sigma_i\rangle_{N,B}(\beta)=\frac{\sum_\sigma \sigma_i e^{-\beta H_{N,B}(\sigma)}}{\sum_\sigma e^{-\beta H_{N,B}(\sigma)}}, \tag{8}$$

and the average magnetization $m_{N,B}(\beta)$ is equal to the same quantity since

$$m_{N,B}(\beta)=\frac{1}{N}\sum_i\langle\sigma_i\rangle_{N,B}(\beta)=\langle\sigma_i\rangle_{N,B}(\beta),$$

where in the last formulation i can be any site in the periodic Ising model due to the translational invariance of the system.

3.2.1. Algorithm. The computation of the quantities mentioned above requires the evaluation of the following sums:

$$\sum_\sigma(\sigma_i\sigma_j)e^{-\beta H_N(\sigma)},\quad \sum_\sigma \sigma_i e^{-\beta H_{N,B}(\sigma)}, \tag{9}$$

where i is any site in the first formula while (i,j) is any bond in the second. Both sums can also be represented using tensor networks using the so-called impurity tensor method.

Recall that the 2D periodic statistical Ising model considered here is of size $n\times n$, where $n=2^L$. Without loss of generality, one can assume that the sites i and j in (9) are located inside the 2×2 sublattice at the center of the whole computation domain. Following the same reasoning in section 1, one can represent $\sum_\sigma(\sigma_i\sigma_j)e^{-\beta H_N(\sigma)}$ and $\sum_\sigma \sigma_i e^{-\beta H_{N,B}(\sigma)}$ as tensor networks. The only difference between them and the tensor network of $Z_N(\beta)$ is a single tensor located inside this 2×2 sublattice at the center.

The algorithm for computing these new tensor networks are quite similar and it becomes particularly simple when the modified TNSK algorithm in section 3.1.4 is used. The whole algorithm is illustrated in Figure 8 and here we only highlight the main differences.

- In addition to the iteration invariance of the modified algorithm in section 3.1.4, one also requires that only the four tensors at the center (marked in gray in Figure 8(a)) can be different from the ones used for $Z_N(\beta)$.

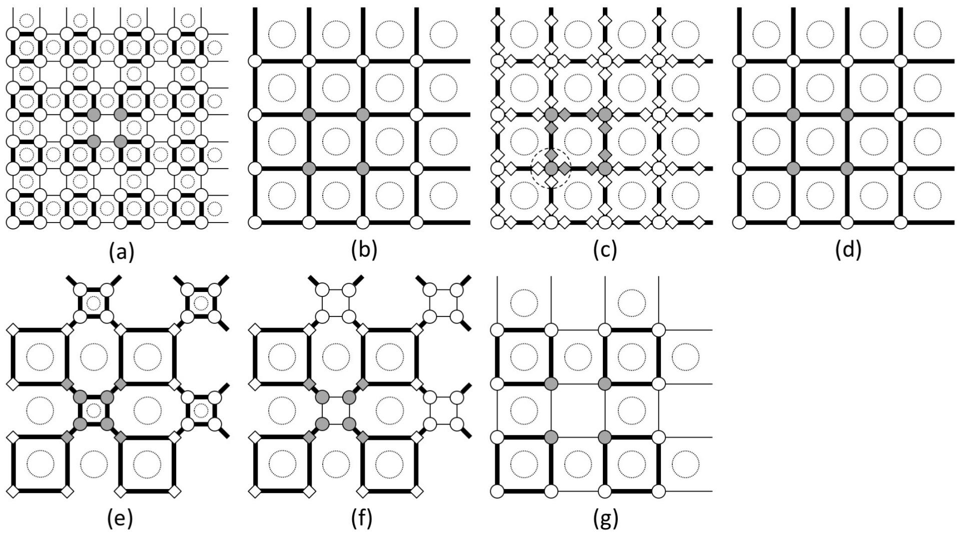

Fig. 8. *Impurity method for computing the spontaneous magnetization and the internal energy per site. The iteration invariance also requires that at the beginning of each iteration only the four tensors near the center can be different from the ones used in $Z_N(\beta)$. These four special tensors are marked in gray. At the end of each iteration, one obtains a coarse-grained tensor network that also satisfies this condition.*

- Because the four special tensors are at the center at the tensor network at level ℓ, after contraction there are exactly four special tensors at the center of the tensor network at level $\ell+1$ (marked in gray in Figure 8(b)). The rest are identical to those used for $Z_N(\beta)$.
- In Figure 8(b) and (c), the $UU^\top T$-projections at the four surrounding edges of the center plaquette are computed from the four special corner tensors. The resulting orthogonal U matrices are marked in gray as well. The $UU^\top T$-projection at all other edges is inherited from the algorithm for the partition function $Z_N(\beta)$. When contracting the tensor at each vertex with its four adjacent orthogonal (diamond) matrices (see Figure 8(c) and (d)), this ensures that only the four tensors at the center are different from those used for $Z_N(\beta)$.
- In the structure-preserving skeletonization step for the $(1,1)_2$ plaquettes (see Figure 8(e) and (f)), only the center $(1,1)_2$ plaquette is different from the one appeared in $Z_N(\beta)$. Therefore, this is the only one that requires an extra structure-preserving skeletonization computation.
- When contracting the tensors at the corners of the $(1,1)_2$ plaquettes to get back the 4-tensors in Figure 8(g), again only the four tensors at the center (marked in gray) are different. This ensures that the tensor network at the beginning of the next iteration satisfies the iteration invariance mentioned above.

At each iteration of this impurity method, the algorithm performs a few extra $UU^\top T$-projection and one extra structure-preserving skeletonization for the $(1,1)_2$ plaquette at the center. As a result, once the computational result for $Z_N(\beta)$ is ready the extra cost from the impurity method is comparable to that for computing the partition function.

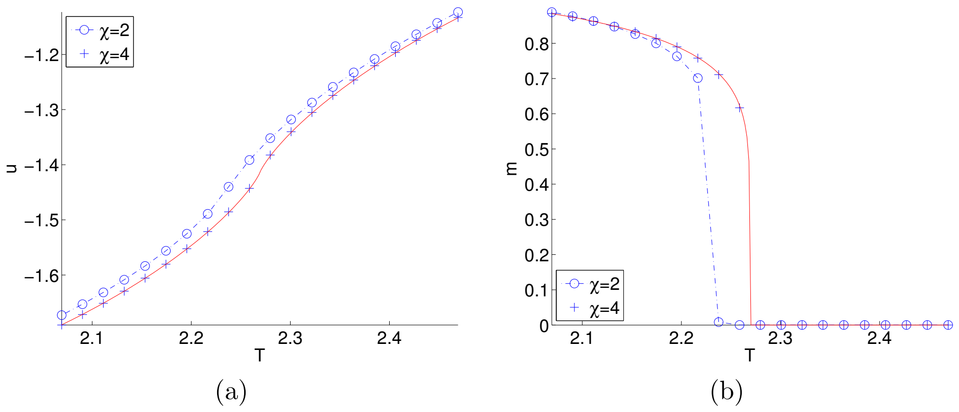

FIG. 9. *Numerical results for computing the observables using the impurity method for* $\chi = 2, 4$. (a) *Internal energy.* (a) *Average magnetization.*

3.2.2. Numerical results. For the internal energy $u_N(\beta)$, we denote by $\tilde{u}_N(\beta)$ its TNSK approximation. When N approaches infinity, the limit

$$u(\beta) = \lim_{N\to\infty} u_N(\beta)$$

is given analytically, for example, in [6]. Therefore for N sufficiently large, $u(\beta)$ serves as a good benchmark for measuring the accuracy of the TNSK algorithm.

For the averaged magnetization, let us denote by $\tilde{m}_{N,B}(\beta)$ the TNSK approximation of $m_{N,B}(\beta)$. For the 2D statistical Ising model, the spontaneous magnetization $m_+(\beta)$ is defined as

$$m_+(\beta) = \lim_{B\to 0^+} \lim_{N\to\infty} m_{N,B}(\beta)$$

and this can be written down analytically as well [6, 19]. When B is a small positive number, by setting N to be sufficiently large, one can treat $m_+(\beta)$ as a good approximation of $m_{N,B}(\beta)$ and use it as a benchmark for measuring the accuracy of $\tilde{m}_{N,B}(\beta)$.

Figure 9(a) shows the computed internal energy per site $\tilde{u}_N(\beta)$ for $\chi = 2, 4$ along with $u(\beta)$. On the right, Figure 9(b) gives the computed average magnetization $\tilde{m}_{N,B}(\beta)$ along with the spontaneous magnetization $m_+(\beta)$. Though the computation with $\chi = 2$ has a significant error, it does exhibit the phase-transition clearly. Once χ is increased to 4, the numerical results and the exact curve match very well.

3.3. Extension to disordered systems. The tensor network skeletonization algorithm can also be extended easily to disordered systems and we briefly sketch how this can be done. For example, consider the 2D Edwards–Anderson spin-glass model (see [8], for example) where the spins are arranged geometrically in the same fashion as the classical Ising model but each edge (i, j) is associated with a parameter J_{ij}. For a fixed realization of $J \equiv \{J_{ij}\}$, the partition function is given by

$$Z_{N,J}(\beta) = \sum_{\sigma} e^{-\beta H_{N,J}(\sigma)}, \quad H_{N,J}(\sigma) = -\sum_{(ij)} J_{ij}\sigma_i\sigma_j.$$

At a fixed realization of J_{ij}, the order parameter of the model is defined as

$$q_{N,J}(\beta) = \frac{1}{N}\sum_{i=1}^{N} \langle\sigma_i\rangle_{N,J}^2(\beta), \quad \langle\sigma_i\rangle_{N,J}(\beta) = \frac{\sum_\sigma \sigma_i e^{-\beta H_{N,J}(\sigma)}}{\sum_\sigma e^{-\beta H_{N,J}(\sigma)}}.$$

The computation of the order parameter $q_N(\beta)$ first requires the evaluation of $Z_{N,J}(\beta)$. Similar to the standard Ising model, this can be represented with a tensor network. The TNSK algorithm remains the same, except that the computation at each plaquette has to be performed separately since the system is not translationally invariant anymore. The computational complexity for $Z_{N,J}(\beta)$ then scales like $O\left((\chi^{12} + n_i \cdot \chi^{10}) \cdot N\right)$, where N is the number of spins.

It also requires the evaluation of $\sum_\sigma \sigma_i e^{-\beta H_J(\sigma)}$ for each i. The discussion in section 3.2 shows that for each i it takes $O\left((\chi^{12} + n_i \cdot \chi^{10}) \cdot \log N\right)$ steps, since most of the computation of $Z_{N,J}(\beta)$ can be reused. Therefore, the computation of $\langle\sigma_i\rangle_{N,J}^2(\beta)$ for all spins i takes $O\left((\chi^{12} + n_i \cdot \chi^{10}) \cdot N\log N\right)$ steps. Putting this and the cost of evaluating $Z_{N,J}(\beta)$ together shows that the computation of the order parameter $q_{N,J}(\beta)$ can be carried out in $O\left((\chi^{12} + n_i \cdot \chi^{10}) \cdot N\log N\right)$ steps.

4. TNSK for 3D statistical Ising model.

In this section, we describe how to extend the tensor network skeletonization algorithm to the 3D statistical Ising model. One key feature that has not been emphasized is that TNSK preserves the Cartesian structure of the problem. This allows for a natural generalization to 3D systems. Let us consider a 3D periodic statistical Ising model on an $n \times n \times n$ Cartesian grid. $N = n^3$ is the number of total spins and we assume without loss of generality $n = 2^L$ for an integer L.

4.1. Partition function.

The partition function $Z_N(\beta)$ can be represented with a tensor network $(V^0, E^0, \{T^i\}_{i\in V^0})$, where V^0 is the set of vertices of the Cartesian grid, the edge set E^0 contains the edges between two adjacent sites in the x, y, and z directions, and T^i is a 6-tensor at site i. This gives rise to an $n \times n \times n$ array of small cubes, each with its eight vertices in V_0. If a cube has vertex $i = (i_1, i_2, i_3)$ at its lower-left-front corner, then we shall index this cube with i as well. We refer to the cubes with index equal to $(0,0,0)$ modulus 2 as $(0,0,0)_2$ cubes and those with index equal to $(1,1,1)$ modulus 2 as $(1,1,1)_2$ cubes. As with the 2D case, we let χ be a predetermined upper bound for the bond dimension and without loss of generality one can assume that $\chi_e \approx \chi$ for each $e \in E^0$.

4.1.1. Algorithm.

The TNSK algorithm consists of a sequence of coarse-graining iterations. At the beginning of each iteration (except the 0th iteration), we require the following iteration invariance to hold:

- For each of the $(0,0,0)_2$ and $(1,1,1)_2$ cubes, the short-range entanglement has already been eliminated.

At the beginning of the ℓth iteration, one holds a tensor network $(V^\ell, E^\ell, \{T^i\}_{i\in V_\ell})$ at level ℓ with $(n/2^\ell) \times (n/2^\ell) \times (n/2^\ell)$ vertices. The ℓth iteration consists of the following steps:

1. Contract the tensors at the eight vertices of each $(0,0,0)_2$ cube into a single tensor (see Figure 10(b)). The $(1,1,1)_2$ cubes are stretched and this results a new tensor network that contains 1/8 of the vertices. The tensors at the new vertices are identical and the bond dimension of the new edges is equal to χ^4 (shown with bold lines in the figure). Similar to the 2D case, the short-range entanglements at level ℓ do not survive to level $\ell + 1$ due to the

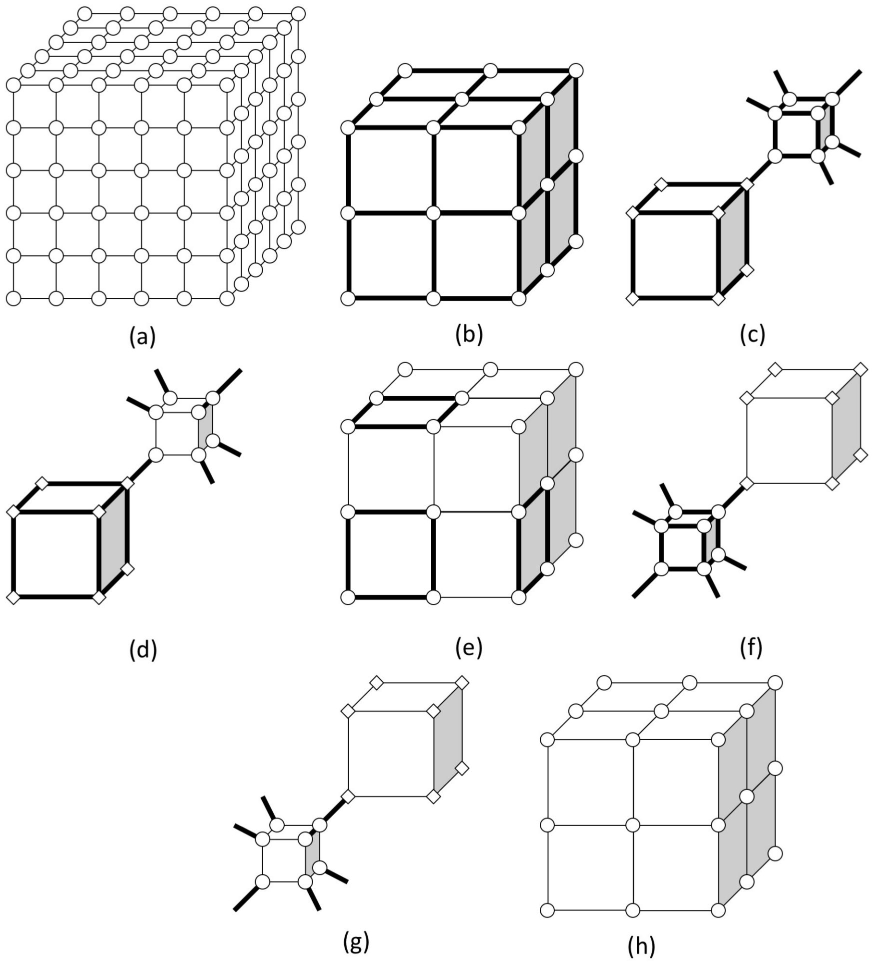

FIG. 10. *A single iteration of the tensor network skeletonization algorithm. The starting point is a tensor network with bond dimension χ and short-range entanglement removed in $(0,0,0)_2$ and $(1,1,1)_2$ cubes. The final point is a coarse-grained tensor network with $1/8$ vertices (tensors). This coarse-grained tensor also has bond dimensions equal to χ and has short-range entanglement removed for the (larger) $(0,0,0)_2$ and $(1,1,1)_2$ cubes.*

iteration invariance. However, there are short-range entanglements for the cubes at level $\ell + 1$. The key task is to remove some of these short-range entanglements and reduce the bond dimension back to χ.

2. At each vertex i in Figure 10(b), denote the tensor by T^i_{abcdef}, where a, b, c, d, e, and f are the left, right, bottom, top, front, and back edges, respectively. By invoking three $UU^\mathsf{T}T$-projection steps (one for each of the left, bottom, and front edges), one effectively inserts two orthogonal (diamond) matrices in each of these edges. At each vertex i, further merge the tensor T^i_{abcdef} with the six adjacent orthogonal (diamond) matrices. This step does not change the topology of the tensor network but the T^i tensor has been modified.
3. For each $(1,1,1)_2$ cube in Figure 10(b), apply the UR-projection to the 6-tensor at each of its corners. Here the three edges adjacent to the cube are grouped together. Notice that the round R tensors are placed close to the

$(1,1,1)_2$ cube. This projection step keeps only the top χ^3 singular values, i.e., the bond dimension of the diagonal edges is equal to χ^3. The resulting graph is given in Figure 10(c).

4. In this key step, apply structure-preserving skeletonization to each of the $(1,1,1)_2$ cubes. The details of this procedure will be provided in section 4.1.2. The resulting $(1,1,1)_2$ plaquette has short-range entanglement removed and the bond dimensions of its 12 surrounding edges is reduced from χ^4 to χ (see Figure 10(d)). One then merges back the UR-projections at its eight corners. The resulting tensor network in Figure 10(e) is similar to the one in Figure 10(b) but the short-range entanglements in the $(1,1,1)_2$ plaquettes are now removed.
5. Now repeat the previous two steps for the $(0,0,0)_2$ plaquettes. This is illustrated in Figure 10(f), (g), and (h). The resulting tensor network has short-range entanglement removed in both $(0,0,0)_2$ and $(1,1,1)_2$ plaquettes and the bond dimension of the edges is reduced back to χ from χ^4.

This finishes the ℓth iteration. At this point, one obtains a new tensor network denoted by $(V^{\ell+1}, E^{\ell+1}, \{T^i\}_{i\in V_{\ell+1}})$ that is a self-similar and coarse-grained version of $(V^\ell, E^\ell, \{T^i\}_{i\in V_\ell})$. This network satisfies the iteration invariance and can serve as the starting point of the next iteration of the algorithm.

The last tensor network $(V^L, E^L, \{T^i\}_{i\in V_L})$ contains only a single 6-tensor with the left and right edges identified and similarly for the bottom/top edges and front/back edges. Contracting this final tensor gives an approximation for the partition function. Similar to the 2D case, one can also introduce a modified version of this algorithm by removing short-range entanglement for the $(1,1,1)_2$ cubes.

4.1.2. Short-range entanglement removal via structure-preserving skeletonization. For the 3D cubes, the structure-preserving skeletonization procedure is similar to the one introduced for the 2D plaquette in section 3.1.2. This procedure is illustrated in Figure 11 with the eight corner 4-tensors denoted by P^{000}, P^{100}, P^{010}, P^{110}, P^{001}, P^{101}, P^{011}, and P^{111}.

Instead of replacing the eight corner 4-tensors of the gray cube simultaneously, this procedure considers the 12 interior edges one by one and inserts within each edge two tensors of size $\chi^4\times\chi$. Starting from the bottom front edge, the procedure seeks two 2-tensors A^0 and A^1 of size $\chi^4\times\chi$ so that the 8-tensor of the new $(1,1,1)_2$ cube after the insertion approximates the original 8-tensors (see Figure 11(a)).

1. Merge the two left tensors P^{000}, P^{001}, P^{010}, and P^{011} into a 5-tensor P^0 and merge the four right tensors into a 5-tensor P^1. After that, the criteria for choosing A^0 and A^1 are given in Figure 11(b), where the two boundary edges have bond dimension equal to $(\chi^3)^4=\chi^{12}$.
2. Since the bond dimensions of the two edges between P^0 and P^1 are to be reduced to χ, this implies that the bond dimensions of the two boundary edges can be reduced to χ^4 instead of χ^{12}. As a result, one can perform the $UU^\mathsf{T}T$-projection to both P^0 and P^1. This gives rise to the criteria for A^0 and A^1 in Figure 11(c).
3. Remove the two tensors U^0 and U^1 at the two endpoints, contract U^0 with P^0 to get a 3-tensor R^0, and contract U^1 with P^1 to get R^1. The approximation criteria for A^0 and A^1 can now be written in terms of R^0 and R^1 as in Figure 11(d).
4. Finally, contracting the three other edges between R^0 and R^1 results in a new 3-tensor T. The approximation criteria for A^0 and A^1 now takes the form in Figure 11(e). This is now exactly the setting of the loop skeletonization

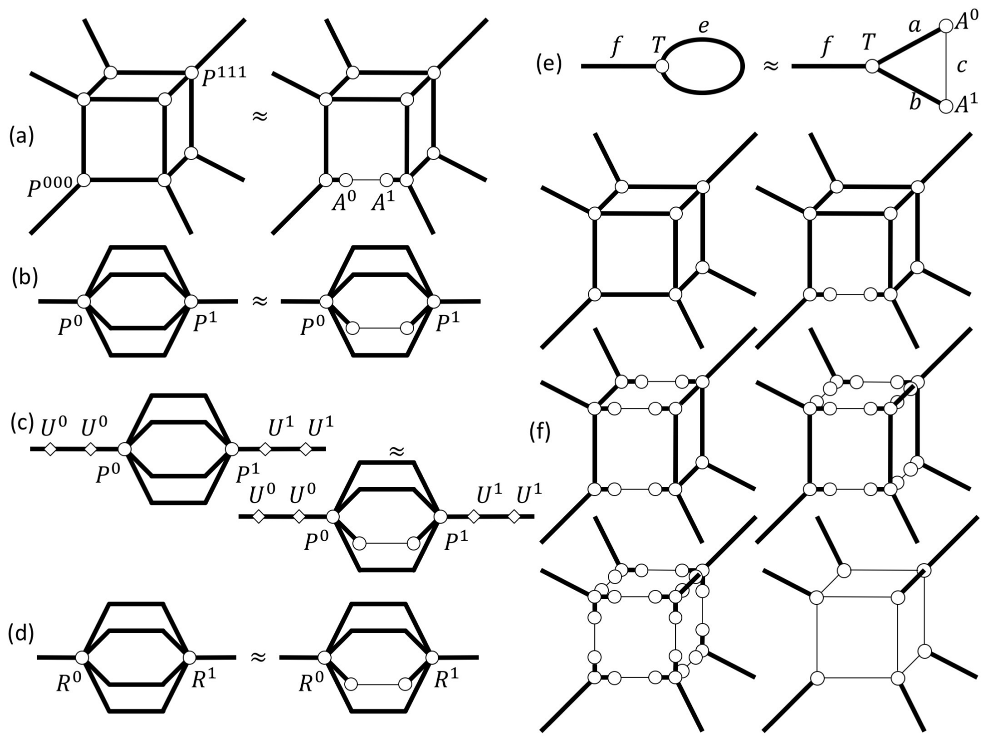

FIG. 11. *The structure-preserving skeletonization removes the short-range entanglement within a* $(1,1,1)_2$ *or* $(0,0,0)_2$ *cube.*

and can be solved using the alternating least square algorithm proposed in section 2.2.

At this point, two tensors A^0 and A^1 are successfully inserted into the bottom front edge. One can repeat this process also for the other three edges in the x direction. Once this is done, we repeat this for the edges in the y direction and then for the edges in the z direction. At this point, there are in total 24 tensors inserted in the 12 surrounding edges of the cube. Finally, merging each of the corner tensors with its three adjacent tensors gives the desired approximation (see Figure 11(f) for the whole process).

4.1.3. Numerical results. The critical temperature of the 3D statistical Ising model is $T_c \approx 4.5115$ but the free energy per site is not known explicitly. For a 3D periodic Ising model on a $2^6 \times 2^6$ lattice, Figure 12 shows the free energy per site obtained through TNSK for $\chi = 2$ at different temperatures near the T_c. The obtained values of the free energy are close to the results obtained from other calculations using HOTRG or Monte Carlo calculations.

4.2. Extensions. Similar to the 2D case, the 3D algorithm can be used to compute the average magnetization and the internal energy per site. When representing these quantities through tensor networks, one finds that only the eight tensors at the center of the computational domain are different from those used in the partition function calculation. Therefore, the impurity tensor method can be applied as expected and the extra computational cost grows like $O(\log N)$ for any fixed χ.

For disordered systems, the same discussion for the 2D systems applies. For example, for computing the order parameter of the 3D Edwards–Anderson model,

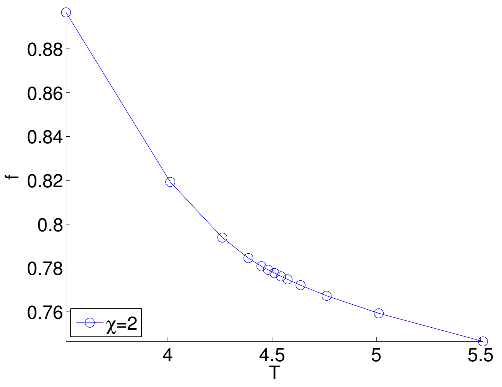

FIG. 12. *The free energy per site obtained through TNSK at temperatures around T_c for $\chi = 2$ for the 3D periodic statistical Ising model.*

one only needs to perform one impurity tensor computation for each site and thus the overall complexity grows like $O(N \log N)$ for any fixed χ.

5. Ground state for quantum Ising models. In this section, we briefly explain how to use TNSK to efficiently represent the ground states of certain quantum many body systems. Consider, for example, a 1D periodic quantum Ising model. One can represent the ground state up to a constant factor using the Euclidean path integrals [1]. After some preliminary tensor manipulations, this turns into a tensor network that is periodic in the spatial dimension and semi-infinite in the imaginary time dimension (see Figure 13(a)).

In the same fashion that the TNR gives rise to the MERA [4] for the ground state, TNSK generates a new representation of the ground state as well. Illustrated in Figure 13, this process consists of the following steps:

1. First, contract each group of 2×2 tensors (see Figure 13(b)). The new edges marked with bold lines have bond dimension equal to χ^2.
2. Perform the structure-preserving skeletonization to all $(0,0)_2$ and $(1,1)_2$ plaquettes to remove the short-range entanglements and reduce the bond dimension back to χ. Notice that after the structure-preserving skeletonization, the resulting tensors at the bottom level are different from the ones above due to their adjacency to the boundary. These special bottom level tensors are marked in gray (see Figure 13(c)).
3. Repeat this process for the remaining tensors above the bottom level. Contracting each group of 2×2 tensors results in a tensor network illustrated in Figures 13(d) and 13(e).
4. One can repeat this process until reaching a half-infinite string of identical matrices. By extracting its top eigenvector, one can reduce this (up to a constant factor) to a 1-tensor at the top (see Figure 13(f)).

The final product is a hierarchical structure shown in Figure 13(f). Though somewhat different from MERA, this new structure also has the capability to represent strongly entangled 1D quantum systems.

For the 2D periodic quantum Ising model, the ground state can be represented via a Euclidean path integral with a 3D tensor network which is periodic in the x and y directions but semi-infinite in the imaginary time direction. The above algorithm (with necessary modifications for the 3D TNSK) can be applied to this tensor network and the result is a hierarchical structure shown in Figure 14.

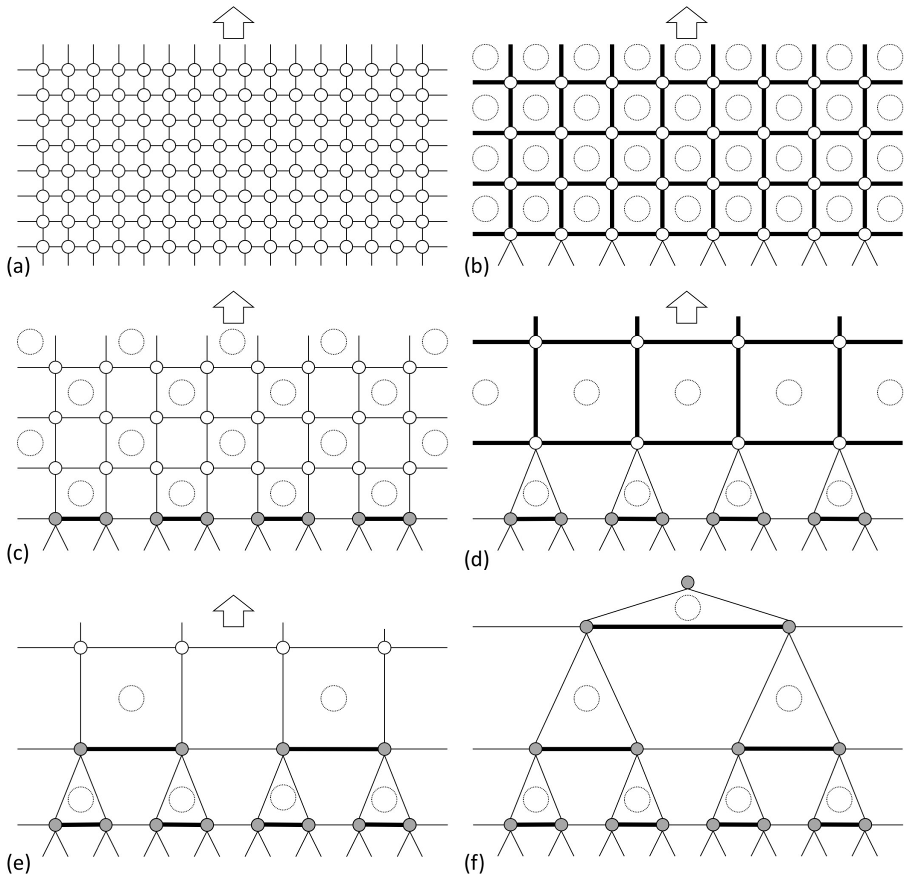

FIG. 13. *When applied to a Euclidean path integral formulation, TNSK yields a new representation of the ground state of* 1*D quantum Ising model.*

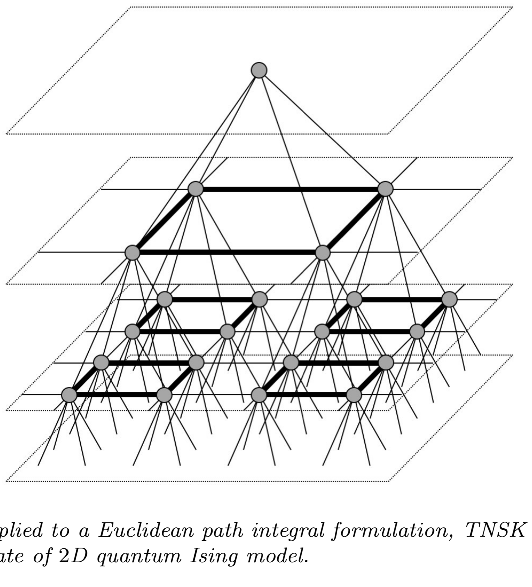

FIG. 14. *When applied to a Euclidean path integral formulation, TNSK yields a new representation of the ground state of* 2*D quantum Ising model.*

6. Conclusion.

This paper introduced TNSK as a new coarse-graining process for the numerical computation of tensor networks. At the heart of TNSK is a new structure-preserving skeletonization procedure that removes short-range entanglement effectively.

As to future work, an immediate task is to investigate other algorithms for the structure-preserving skeletonization problem (4) and (5). The alternating least square algorithm adopted here works quite well in practice. However, it would be interesting to understand why and also to consider other alternatives without using the somewhat artificial regularization parameter.

Most TNSK algorithms introduced here are presented in their simplest forms in order to illustrate the main ideas. They are not necessarily the most efficient implementations in practice. For example, in the TNSK algorithm for partition functions, one performs the contractions over all directions simultaneously and then applies the $UU^\mathsf{T}T$-projections to these directions. However, in practice, it is much more efficient to iterate over the directions and apply a $UU^\mathsf{T}T$-projection right after the contraction of each direction.

Acknowledgments. The author thanks Lin Lin, Xiao-liang Qi, Tao Xiang, Zhiyuan Xie, and Wotao Yin for stimulating discussions on various parts of this work.

REFERENCES

[1] G. Evenbly, *Algorithms for Tensor Network Renormalization*, arXiv:1509.07484, 2015.

[2] G. Evenbly and G. Vidal, *Algorithms for entanglement renormalization*, Phys. Rev. B, 79 (2009), 144108.

[3] G. Evenbly and G. Vidal, *Tensor network renormalization*, Phys. Rev. Lett., 115 (2015), 180405.

[4] G. Evenbly and G. Vidal, *Tensor network renormalization yields the multiscale entanglement renormalization ansatz*, Phys. Rev. Lett., 115 (2015), 200401.

[5] Z.-C. Gu and X.-G. Wen, *Tensor-entanglement-filtering renormalization approach and symmetry-protected topological order*, Phys. Rev. B, 80 (2009), 155131.

[6] K. Huang, *Statistical Mechanics*, 2nd ed., John Wiley & Sons, New York, 1987.

[7] M. Levin and C. P. Nave, *Tensor renormalization group approach to two-dimensional classical lattice models*, Phys. Rev. Lett., 99 (2007), 120601.

[8] H. Nishimori, *Statistical Physics of Spin Glasses and Information Processing*, Internat. Ser. Monogr. Phys. 111, Oxford University Press, New York, 2001.

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

[10] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac, *Matrix product state representations*, Quantum Info. Comput., 7 (2007), pp. 401–430.

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

[12] U. Schollwöck, *The density-matrix renormalization group*, Rev. Mod. Phys., 77 (2005), pp. 259–315.

[13] F. Verstraete and J. I. Cirac, *Renormalization Algorithms for Quantum-Many Body Systems in Two and Higher Dimensions*, arXiv:cond-mat/0407066, 2004.

[14] G. Vidal, *Entanglement renormalization*, Phys. Rev. Lett., 99 (2007), 220405.

[15] G. Vidal, *Class of quantum many-body states that can be efficiently simulated*, Phys. Rev. Lett., 101 (2008), 110501.

[16] S. R. White, *Density matrix formulation for quantum renormalization groups*, Phys. Rev. Lett., 69 (1992), pp. 2863–2866.

[17] Z. Y. Xie, J. Chen, M. P. Qin, J. W. Zhu, L. P. Yang, and T. Xiang, *Coarse-graining renormalization by higher-order singular value decomposition*, Phys. Rev. B, 86 (2012), 045139.

[18] Z. Y. Xie, H. C. Jiang, Q. N. Chen, Z. Y. Weng, and T. Xiang, *Second renormalization of tensor-network states*, Phys. Rev. Lett., 103 (2009), 160601.

[19] C. N. Yang, *The spontaneous magnetization of a two-dimensional ising model*, Phys. Rev., 85 (1952), pp. 808–816.

[20] S. Yang, Z.-C. Gu, and X.-G. Wen, *Loop Optimization for Tensor Network Renormalization*, arXiv:1512.04938, 2015.

[21] H. H. Zhao, Z. Y. Xie, Q. N. Chen, Z. C. Wei, J. W. Cai, and T. Xiang, *Renormalization of tensor-network states*, Phys. Rev. B, 81 (2010), 174411.