Tensor network states and algorithms in the presence of a global U(1) symmetry

Sukhwinder Singh, Robert N. C. Pfeifer, and Guifre Vidal

Department of Physics, The University of Queensland, Brisbane QLD 4072, Australia
(Received 12 October 2010; published 15 March 2011)

Tensor network decompositions offer an efficient description of certain many-body states of a lattice system and are the basis of a wealth of numerical simulation algorithms. In a recent paper [Phys. Rev. A **82**, 050301 (2010)] we discussed how to incorporate a global internal symmetry, given by a compact, completely reducible group $\mathcal{G}$, into tensor network decompositions and algorithms. Here we specialize to the case of Abelian groups and, for concreteness, to a U(1) symmetry, associated, e.g., with particle number conservation. We consider tensor networks made of tensors that are invariant (or covariant) under the symmetry, and explain how to decompose and manipulate such tensors in order to exploit their symmetry. In numerical calculations, the use of U(1)-symmetric tensors allows selection of a specific number of particles, ensures the exact preservation of particle number, and significantly reduces computational costs. We illustrate all these points in the context of the multiscale entanglement renormalization *Ansatz*.

I. INTRODUCTION

Tensor networks are becoming increasingly popular as a tool to represent wave functions of quantum many-body systems. Their success is based on the ability to efficiently describe the ground state of a broad class of local Hamiltonians on the lattice. Tensor network states are used both as a variational *Ansatz* to numerically approximate ground states and as a theoretical framework to characterize and classify quantum phases of matter.

Examples of tensor network states for one-dimensional systems include the matrix product state[1–3] (MPS), which results naturally from both Wilson's numerical renormalization group[4] and White's density-matrix renormalization group[5–8] (DMRG) and is also used as a basis for simulation of time evolution, e.g., with the time evolving block decimation (TEBD)[9–11] algorithm and variations thereof, often collectively referred to as time-dependent DMRG;[9–14] the tree tensor network[15] (TTN), which follows from coarse-graining schemes where the spins are blocked hierarchically; and the multiscale entanglement renormalization *Ansatz*[16–21] (MERA), which results from a renormalization-group procedure known as entanglement renormalization.[16,21] For two-dimensional (2D) lattices there are generalizations of these three tensor network states, namely projected entangled pair states[22–31] (PEPS), 2D TTN,[32,33] and 2D MERA,[34–40] respectively. As variational *Ansätze*, PEPS and 2D MERA are particularly interesting since they can be used to address large two-dimensional lattices, including systems of frustrated spins[31,40] and interacting fermions,[41–50] where Monte Carlo techniques fail due to the sign problem.

A many-body Hamiltonian $\hat{H}$ may be invariant under certain transformations that form a group of symmetries.[51] The symmetry group divides the Hilbert space of the theory into symmetry sectors labeled by quantum numbers or conserved charges. On a lattice one can distinguish between *space* symmetries, which correspond to some permutation of the sites of the lattice, and *internal* symmetries, which act on the vector space of each site. An example of space symmetry is invariance under translations by some unit cell, which leads to conservation of momentum. An example of internal symmetry

is SU(2) invariance, e.g., spin isotropy in a quantum spin model. An internal symmetry can in turn be *global*, if it transforms the space of each of the lattice sites according to the same transformation (e.g., a spin independent rotation); or *local*, if each lattice site is transformed according to a different transformation (e.g., a spin dependent rotation), as it is in the case of gauge symmetric models. A global internal SU(2) symmetry gives rise to conservation of total spin. By targeting a specific symmetry sector during a calculation, computational costs can often be significantly reduced while explicitly preserving the symmetry. It is therefore not surprising that symmetries play an important role in numerical approaches.

In Ref. 52 we described a formalism for incorporating global internal symmetries into a generic tensor network algorithm. Both Abelian and non-Abelian symmetries were considered. The purpose of this paper is to address, at a pedagogical level, the implementation of Abelian symmetries into tensor networks. We will also discuss several practical aspects of the exploitation of Abelian symmetries not covered in Ref. 52. For concreteness we will concentrate on the U(1) symmetry, but extending our results to any Abelian group is straightforward. A similar analysis of non-Abelian groups will be considered in Ref. 53.

In tensor network approaches, the exploitation of global internal symmetries has a long history, especially in the context of MPSs. Both Abelian and non-Abelian symmetries have been thoroughly incorporated into DMRG code and have been exploited to obtain computational gains.[2,5,14,54–62] Symmetries have also been used in more recent proposals to simulate time evolution with MPSs.[10–14,63–68]

When considering symmetries, it is important to notice that an MPS is a trivalent tensor network. That is, in an MPS each tensor has at most three indices. The Clebsch-Gordan coefficients[51] (or coupling coefficients) of a symmetry group are also trivalent, and this makes incorporating the symmetry into an MPS by considering symmetric tensors particularly simple. In contrast, tensor network states with a more elaborate network of tensors, such as MERA or PEPS, consist of tensors having a larger number of indices. In this case a more general formalism is required in order to exploit the symmetry. As

explained in Ref. 52, a generic symmetric tensor can be decomposed into a *degeneracy* part, which contains all degrees of freedom not determined by symmetry, and a *structural* part, which is completely determined by symmetry and can be further decomposed as a trivalent network of Clebsch-Gordan coefficients.

The use of symmetric tensors in more complex tensor networks has also been discussed in Refs. 69,70. In particular, Ref. 69 has shown that under convenient conditions (injectivity), a PEPS that represents a symmetric state can be represented with symmetric tensors, generalizing similar results for MPSs obtained in Ref. 61. Notice that these studies are not concerned with how to decompose symmetric tensors so as to computationally protect or exploit the symmetry. On the other hand, exploitation of U(1) symmetry for computational gain in the context of PEPS was reported in Ref. 70, although no implementation details were provided. Finally, several aspects of *local* internal symmetries in tensor network algorithms have been addressed in Refs. 71–74.

The paper is organized in sections as follows. Section II contains a review of the tensor network formalism and introduces the nomenclature and diagrammatical representation of tensors used in the rest of the paper. It also describes a set $\mathcal{P}$ of primitives for manipulating tensor networks, consisting of manipulations that involve a single tensor (permutation, fusion, and splitting of the indices of a tensor) and matrix operations (multiplication and factorization).

Section III reviews basic notions of representation theory of the Abelian group U(1). The action of the group is analyzed first on a single vector space, where U(1)-symmetric states and U(1)-invariant operators are decomposed in a compact, canonical manner. This canonical form allows us to identify the degrees of freedom which are not constrained by the symmetry. The action of the group is then also analyzed on the tensor product of two vector spaces and, finally, on the tensor product of a finite number of vector spaces.

Section IV explains how to incorporate the U(1) symmetry into a generic tensor network algorithm, by considering U(1)-invariant tensors in a canonical form, and by adapting the set $\mathcal{P}$ of primitives for manipulating tensor networks. These include the multiplication of two U(1)-invariant matrices in their canonical form, which is at the core of the computational savings obtained by exploiting the symmetry in tensor network algorithms.

Section V illustrates the practical exploitation of the U(1) symmetry in a tensor network algorithm by presenting MERA calculations of the ground state and low-energy states of two quantum spin chain models. Section VI contains some conclusions.

The canonical form offers a more compact description of U(1)-invariant tensors, and leads to faster matrix multiplications and factorizations. However, there is also an additional cost associated with maintaining an invariant tensor in its canonical form while reshaping (fusing and/or splitting) its indices. In some situations, this cost may offset the benefits of using the canonical form. In the Appendix we discuss a scheme to lower this additional cost in tensor network algorithms that are based on iterating a repeated sequence of transformations. This is achieved by identifying, in the manipulation of a tensor, operations which only depend on the symmetry. Such

operations can be *precomputed* once at the beginning of a simulation. Their result, stored in memory, can be reused at each iteration of the simulation. The Appendix describes two such specific precomputation schemes.

II. REVIEW: TENSOR NETWORK FORMALISM

In this section we review background material concerning the formalism of tensor networks, without reference to symmetry. We introduce basic definitions and concepts, as well as the nomenclature and graphical representation for tensors, tensor networks, and their manipulations, that will be used throughout the paper.

A. Tensors

A tensor $\hat{T}$ is a multidimensional array of complex numbers $\hat{T}_{i_1 i_2 \cdots i_k} \in \mathbb{C}$. The *rank* of tensor $\hat{T}$ is the number k of indices. For instance, a rank-0 tensor ($k = 0$) is a complex number. Similarly, rank-1 ($k = 1$) and rank-2 ($k = 2$) tensors represent vectors and matrices, respectively. The *size* of an index i, denoted $|i|$, is the number of values that the index takes, $i \in \{1, 2, \ldots, |i|\}$. The size of a tensor $\hat{T}$, denoted $|\hat{T}|$, is the number of complex numbers it contains, namely $|\hat{T}| = |i_1| \times |i_2| \times \cdots \times |i_k|$. In this paper we will use the hat, ^, to indicate that an object is a tensor. We include vectors in this convention, writing their components as, e.g., $\hat{\Psi}_i$, although for simplicity we will omit the hat when a vector is written in bra or ket form, e.g., $|\Psi\rangle$.

It is convenient to use a graphical representation of tensors, as introduced in Fig. 1, where a tensor $\hat{T}$ is depicted as a circle (more generally some shape, e.g., a square) and each of its indices is represented by a line emerging from it. In order to specify which index corresponds to which emerging line, we follow the prescription that the lines corresponding to indices $\{i_1, i_2, \ldots, i_k\}$ emerge in counterclockwise order. Unless stated otherwise, the first index will correspond to the line emerging at nine o'clock (or the first line encountered while proceeding counterclockwise from nine o'clock).

Two elementary ways in which a tensor $\hat{T}$ can be transformed are by *permuting* and *reshaping* its indices. A *permutation* of indices corresponds to creating a new tensor $\hat{T}'$ from $\hat{T}$ by simply changing the order in which the indices appear, e.g.,
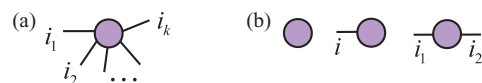
$$(\hat{T}')_{acb} = \hat{T}_{abc}. \tag{1}$$

FIG. 1. (Color online) (a) Graphical representation of a tensor $\hat{T}$ of rank k and components $\hat{T}_{i_1 i_2 \cdots i_k}$. The tensor is represented by a shape (circle) with k emerging lines corresponding to the k indices $i_1, i_2, \ldots, i_k$. Notice that the indices emerge in counterclockwise order. (b) Graphical representation of tensors with rank $k = 0, 1$, and 2, corresponding to a complex number $c \in \mathbb{C}$, a vector $|v\rangle \in \mathbb{C}^{|i|}$, and a matrix $\hat{M} \in \mathbb{C}^{|i_1| \times |i_2|}$, respectively.
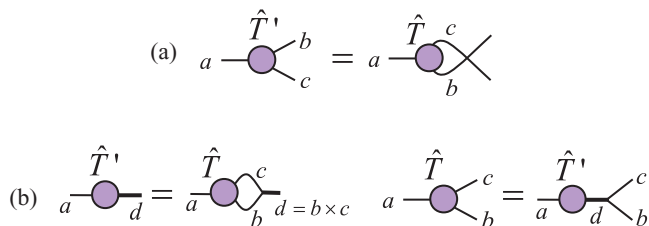
FIG. 2. (Color online) Transformations of a tensor: (a) Permutation of indices b and c. (b) Fusion of indices b and c into $d = b \times c$; splitting of index $d = b \times c$ into b and c.

On the other hand, a tensor $\hat{T}$ can be *reshaped* into a new tensor $\hat{T}'$ by "fusing" and/or "splitting" some of its indices. For instance, in

$$(\hat{T}')_{ad} = \hat{T}_{abc}, \quad d = b \times c, \tag{2}$$

tensor $\hat{T}'$ is obtained from tensor $\hat{T}$ by fusing indices $b \in \{1, \ldots, |b|\}$ and $c \in \{1, \ldots, |c|\}$ together into a single index d of size $|d| = |b| \cdot |c|$ that runs over all pairs of values of b and c, i.e., $d \in \{(1,1),(1,2),\ldots,(|b|,|c|-1),(|b|,|c|)\}$, whereas in

$$\hat{T}_{abc} = (\hat{T}')_{ad}, \quad d = b \times c, \tag{3}$$

tensor $\hat{T}$ is recovered from $\hat{T}'$ by splitting index d of $\hat{T}'$ back into indices b and c. The permutation and reshaping of the indices of a tensor have a straightforward graphical representation; see Fig. 2.

B. Multiplication of two tensors

Given two matrices $\hat{R}$ and $\hat{S}$ with components $\hat{R}_{ab}$ and $\hat{S}_{bc}$, we can multiply them together to obtain a new matrix $\hat{T}$, $\hat{T} = \hat{R} \cdot \hat{S}$, with components

$$\hat{T}_{ac} = \sum_b \hat{R}_{ab} \hat{S}_{bc}, \tag{4}$$

by summing over or *contracting* index b. The multiplication of matrices $\hat{R}$ and $\hat{S}$ is represented graphically by connecting together the emerging lines of $\hat{R}$ and $\hat{S}$ corresponding to the contracted index, as shown in Fig. 3(a).

Matrix multiplication can be generalized to tensors. For instance, given tensors $\hat{R}$ and $\hat{S}$ with components $\hat{R}_{abcd}$ and
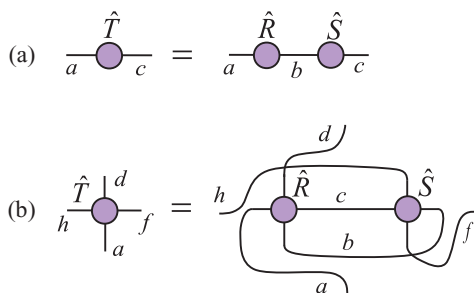

FIG. 3. (Color online) (a) Graphical representation of the matrix multiplication of two matrices $\hat{R}$ and $\hat{S}$ into a new matrix $\hat{T}$ [Eq. (4)]. (b) Graphical representation of an example of the contraction of two tensors $\hat{R}$ and $\hat{S}$ into a new tensor $\hat{T}$ [Eq. (5)].

$\hat{S}_{cfbh}$, we can define a tensor $\hat{T}$ with components $\hat{T}_{hafd}$ given by

$$\hat{T}_{hafd} = \sum_{bc} \hat{R}_{abcd} \hat{S}_{cfbh}. \tag{5}$$

Again the multiplication of two tensors can be graphically represented by connecting together the lines corresponding to indices that are being contracted [indices b and c in Eq. (5)]; see Fig. 3(b).

The multiplication of two tensors can be broken down into a sequence of elementary steps by transforming the tensors into matrices, multiplying the matrices together, and then transforming the resulting matrix back into a tensor. Next we describe these steps for the contraction given in Eq. (5). They are illustrated in Fig. 4.

(1) *Permute* the indices of tensor $\hat{R}$ in such a way that the indices to be contracted, b and c, appear in the last positions and in a given order, e.g., bc; similarly, permute the indices of $\hat{S}$ so that the indices to be contracted, again b and c, appear in the first positions and in the same order bc:

$$(\hat{R}')_{adbc} = \hat{R}_{abcd},$$
$$(\hat{S}')_{bcfh} = \hat{S}_{cfbh}. \tag{6}$$

(2) *Reshape* tensor $\hat{R}'$ into a matrix $\hat{R}''$ by fusing into a single index u all the indices that are not going to be contracted, $u = a \times d$, and into a single index y all indices to be contracted, $y = b \times c$. Similarly, reshape tensor $\hat{S}'$ into a matrix $\hat{S}''$ with indices $y = b \times c$ and $w = f \times h$,

$$(\hat{R}'')_{uy} = (\hat{R}')_{adbc},$$
$$(\hat{S}'')_{yw} = (\hat{S}')_{bcfh}. \tag{7}$$

(3) *Multiply* matrices $\hat{R}''$ and $\hat{S}''$ to obtain a matrix $\hat{T}''$, with components

$$(\hat{T}'')_{uw} = \sum_y (\hat{R}'')_{uy}(\hat{S}'')_{yw}. \tag{8}$$
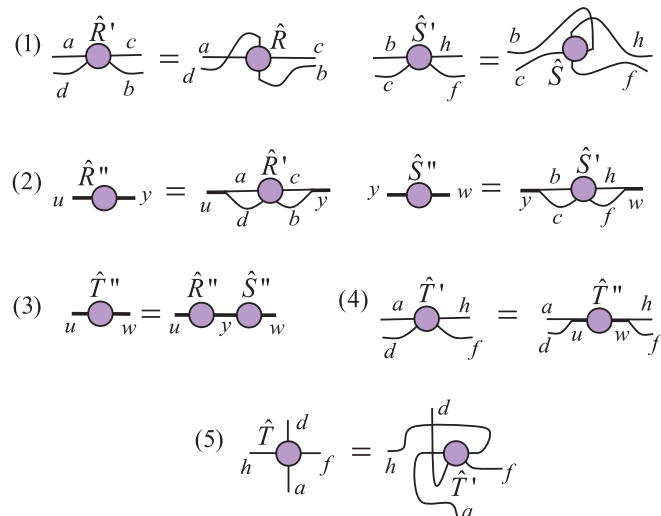

FIG. 4. (Color online) Graphical representations of the five elementary steps (1)–(5) into which one can decompose the contraction of the tensors of Eq. (5).

(4) *Reshape* matrix $\hat{T}''$ into a tensor $\hat{T}'$ by splitting indices $u = a \times d$ and $w = f \times h$,

$$(\hat{T}')_{adfh} = (\hat{T}'')_{uw}. \tag{9}$$

(5) *Permute* the indices of $\hat{T}'$ into the order in which they appear in $\hat{T}$,

$$\hat{T}_{hafd} = (\hat{T}')_{adfh}. \tag{10}$$

We note that breaking down a multiplication of two tensors into elementary steps is not necessary—one can simply implement the contraction of Eq. (5) as a single process. However, it is often more convenient to compose the above elementary steps since, for instance, in this way one can use existing linear algebra libraries for matrix multiplication. In addition, it can be seen that the leading computational cost in multiplying two large tensors is not changed when decomposing the contraction in the above steps. In Sec. IV I this subject will be discussed in more detail for U(1)-invariant tensors.

C. Factorization of a tensor

A matrix $\hat{T}$ can be factorized into the product of two (or more) matrices in one of several canonical forms. For instance, the *singular value decomposition*

$$\hat{T}_{ab} = \sum_{c,d} \hat{U}_{ac} \hat{S}_{cd} \hat{V}_{db} = \sum_{c} \hat{U}_{ac} s_c \hat{V}_{cb} \tag{11}$$

factorizes $\hat{T}$ into the product of two unitary matrices $\hat{U}$ and $\hat{V}$, and a diagonal matrix $\hat{S}$ with non-negative diagonal elements $s_c = \hat{S}_{cc}$ known as the *singular values* of $\hat{T}$; see Fig. 5(a).

On the other hand, the *eigenvalue* or *spectral decomposition* of a square matrix $\hat{T}$ is of the form

$$\hat{T}_{ab} = \sum_{c,d} \hat{M}_{ac} D_{cd} (\hat{M}^{-1})_{db} = \sum_{c} \hat{M}_{ac} \lambda_c (\hat{M}^{-1})_{cb}, \tag{12}$$

where $\hat{M}$ is an invertible matrix whose columns encode the eigenvectors $|\lambda_c\rangle$ of $\hat{T}$,

$$\hat{T}|\lambda_c\rangle = \lambda_c |\lambda_c\rangle, \tag{13}$$

$\hat{M}^{-1}$ is the inverse of $\hat{M}$, and $\hat{D}$ is a diagonal matrix, with the eigenvalues $\lambda_c = \hat{D}_{cc}$ on its diagonal. Other useful factorizations include the LU decomposition, the QR decomposition, etc. We refer to any such decomposition generically as a *matrix factorization*.

A tensor $\hat{T}$ with more than two indices can be converted into a matrix in several ways, by specifying how to join its indices into two subsets. After specifying how tensor $\hat{T}$ is to be regarded as a matrix, we can factorize $\hat{T}$ according to any of the above matrix factorizations, as illustrated in Fig. 5(b) for a singular value decomposition. This requires first permuting and reshaping the indices of $\hat{T}$ to form a matrix, then decomposing the latter, and finally restoring the open indices of the resulting matrices into their original form by undoing the reshapes and permutations.
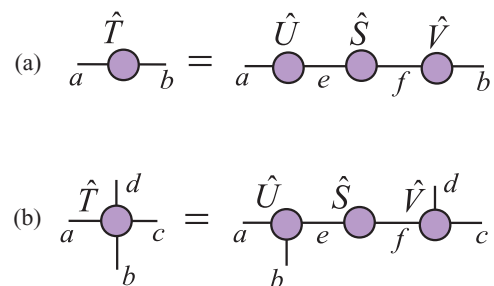

FIG. 5. (Color online) (a) Factorization of a matrix $\hat{T}$ according to a singular value decomposition (11). (b) Factorization of a rank-4 tensor $\hat{T}$ according to one of several possible singular value decompositions.

D. Tensor networks and their manipulation

A *tensor network* $\mathcal{N}$ is a set of tensors whose indices are connected according to a network pattern, e.g., Fig. 6. Given a tensor network $\mathcal{N}$, a single tensor $\hat{T}$ can be obtained by contracting all the indices that connect the tensors in $\mathcal{N}$ [Fig. 6(b)]. Here, the indices of tensor $\hat{T}$ correspond to the open indices of the tensor network $\mathcal{N}$. We then say that the network $\mathcal{N}$ is a tensor network decomposition of $\hat{T}$. One way to obtain $\hat{T}$ from $\mathcal{N}$ is through a sequence of contractions involving two tensors at a time [Fig. 6(c)].

From a tensor network decomposition $\mathcal{N}$ for a tensor $\hat{T}$, another tensor network decomposition for the same tensor $\hat{T}$ can be obtained in many ways. One possibility is to replace two tensors in $\mathcal{N}$ with the tensor resulting from contracting them together, as is done in each step of Fig. 6(c). Another way is to replace a tensor in $\mathcal{N}$ with a decomposition of that tensor (e.g., with a singular value decomposition). In this paper, we will be concerned with manipulations of a tensor network that, as in the case of multiplying two tensors or decomposing a tensor, can be broken down into a sequence of operations from the following list:

(1) Permutation of the indices of a tensor, Eq. (1).
(2) Reshape of the indices of a tensor, Eqs. (2) and (3).
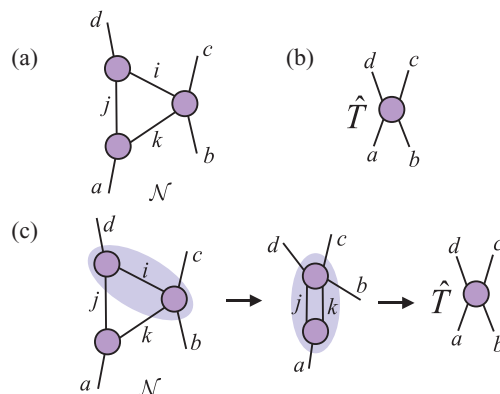(3) Multiplication of two matrices, Eq. (4).

FIG. 6. (Color online) (a) Example of a tensor network $\mathcal{N}$. (b) Tensor $\hat{T}$ of which the tensor network $\mathcal{N}$ could be a representation. (c) Tensor $\hat{T}$ can be obtained from $\mathcal{N}$ through a sequence of contractions of pairs of tensors. Shading indicates the two tensors to be multiplied together at each step.

(4) Decomposition of a matrix [e.g., singular value decomposition (11) or spectral decomposition (12)].

These operations constitute a set $\mathcal{P}$ of *primitive* operations for tensor network manipulations (or, at least, for the type of manipulations we will be concerned with). In Sec. IV we will discuss how this set $\mathcal{P}$ of primitive operations can be generalized to tensors that are symmetric under the action of the group U(1).

E. Tensor network states for quantum many-body systems

As mentioned in the Introduction of the paper, tensor networks are used as a means to represent the wave function of certain quantum many-body systems on a lattice. Let us consider a lattice $\mathcal{L}$ made of L sites, each described by a complex vector space $\mathbb{V}$ of dimension d. A generic pure state $|\Psi\rangle \in \mathbb{V}^{\otimes L}$ of $\mathcal{L}$ can always be expanded as

$$|\Psi\rangle = \sum_{i_1,i_2,\ldots,i_L} \hat{\Psi}_{i_1 i_2 \cdots i_L} |i_1\rangle |i_2\rangle \cdots |i_L\rangle, \quad (14)$$

where $i_s = 1, \ldots, d$ labels a basis $|i_s\rangle$ of $\mathbb{V}$ for site $s \in \mathcal{L}$. Tensor $\hat{\Psi}$, with components $\hat{\Psi}_{i_1 i_2 \cdots i_L}$, contains d^L complex coefficients. This is a number that grows exponentially with the size L of the lattice. Thus the representation of a *generic* pure state $|\Psi\rangle \in \mathbb{V}^{\otimes L}$ is *inefficient*. However, it turns out that an *efficient* representation of *certain* pure states can be obtained by expressing tensor $\hat{\Psi}$ in terms of a tensor network.

Figure 7 shows several popular tensor network decompositions used to approximately describe the ground states of local Hamiltonians H of lattice models in one or two spatial dimensions. The open indices of each of these tensor networks correspond to the indices $i_1, i_2, \ldots, i_L$ of tensor $\hat{\Psi}$. Notice that all the tensor networks of Fig. 7 contain $O(L)$ tensors. If p is the rank of the tensors in one of these tensor networks, and χ is the size of their indices, then the tensor network depends on $O(L\chi^p)$ complex coefficients. For a fixed value of χ this number grows linearly in L, and not exponentially.
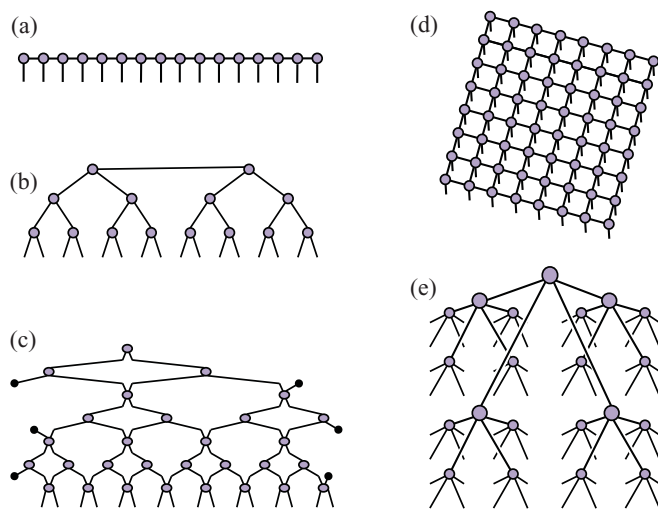

FIG. 7. (Color online) Examples of tensor network states for 1D systems: (a) matrix product state (MPS), (b) tree tensor network (TTN), (c) multiscale entanglement renormalization *Ansatz* (MERA). Examples of tensor network states for 2D systems: (d) projected entangled-pair state PEPS, (e) 2D TTN (2D MERA not depicted).

It therefore does indeed offer an efficient description of the pure state $|\Psi\rangle \in \mathbb{V}^{\otimes L}$ that it represents. Of course only a subset of pure states can be decomposed in this way. Such states, often referred to as tensor network states, are used as variational *Ansätze*, with the $O(L\chi^p)$ complex coefficients as the variational parameters.

Given a tensor network state, a variety of algorithms (see, e.g., Refs. 4–50) are used for tasks such as (i) computation of the expectation value $\langle\Psi|\hat{o}|\Psi\rangle$ of a local observable $\hat{o}$, (ii) optimization of the variational parameters so as to minimize the expectation value of the energy $\langle\Psi|\hat{H}|\Psi\rangle$, or (iii) simulation of time evolution, e.g., $e^{-i\hat{H}t}|\Psi\rangle$. These tasks are accomplished by manipulating tensor networks.

On most occasions, all required manipulations can be reduced to a sequence of primitive operations in the set $\mathcal{P}$ introduced in Sec. II D. Thus in order to adapt the tensor network algorithms of, e.g., Refs. 4–50 to the presence of a symmetry, we only need to modify the set $\mathcal{P}$ of primitive tensor network operations. This will be done in Sec. IV.

F. Tensors as linear maps

A tensor can be used to define a linear map between vector spaces in the following way. First, notice that an index i can be used to label a basis $\{|i\rangle\}$ of a complex vector space $\mathbb{V}^{[i]} \cong \mathbb{C}^{|i|}$ of dimension $|i|$. On the other hand, given a tensor $\hat{T}$ of rank k, we can attach a direction "in" or "out" to each index $i_1, i_2, \ldots, i_k$. This direction divides the indices of $\hat{T}$ into the subset I of *incoming* indices and the subset O of *outgoing* indices. We can then build input and output vector spaces given by the tensor product of the spaces of incoming and outgoing indices,

$$\mathbb{V}^{[\mathrm{in}]} = \bigotimes_{i_l \in I} \mathbb{V}^{[i_l]}, \quad \mathbb{V}^{[\mathrm{out}]} = \bigotimes_{i_l \in O} \mathbb{V}^{[i_l]}, \quad (15)$$

and use tensor $\hat{T}$ to define a linear map between $\mathbb{V}^{[\mathrm{in}]}$ and $\mathbb{V}^{[\mathrm{out}]}$. For instance, if a rank-3 tensor $\hat{T}_{abc}$ has one incoming index $c \in I$ and two outgoing indices $a, b \in O$, then it defines a linear map $\hat{T} : \mathbb{V}^{[c]} \to \mathbb{V}^{[a]} \otimes \mathbb{V}^{[b]}$ given by

$$\hat{T} = \sum_{a,b,c} \hat{T}_{abc} |a\rangle|b\rangle\langle c|. \quad (16)$$

Graphically, we denote the direction of an index by means of an arrow; see Fig. 8(a).

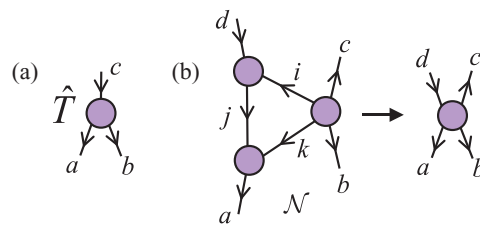FIG. 8. (Color online) (a) Tensor $\hat{T}$ with one incoming index and two outgoing indices, denoted by incoming and outgoing arrows, respectively [Eq. (16)]. (b) A tensor network $\mathcal{N}$ with directed links can be interpreted as a linear map between incoming and outgoing spaces (of the incoming and outgoing indices) obtained by composing the linear maps associated with each of the tensors in $\mathcal{N}$.

By decorating the lines of a tensor network $\mathcal{N}$ with arrows [Fig. 8(b)], this can be regarded as a composition of linear maps—namely, one linear map for each tensor in $\mathcal{N}$. While arrows might be of limited relevance in the absence of a symmetry, they will play an important role when we consider symmetric tensors since they specify how the group acts on each index of a given tensor.

III. REVIEW: REPRESENTATION THEORY OF THE GROUP U(1)

In this section we review basic background material concerning the representation theory of the group U(1). We first consider the action of U(1) on a vector space $\mathbb{V}$, which decomposes into the direct sum of (possibly degenerate) irreducible representations. We then consider vectors of $\mathbb{V}$ that are symmetric (invariant or covariant) under the action of U(1), as well as linear operators that are U(1) invariant. Then we consider the action of U(1) on the tensor product of two vector spaces, and its generalization to the tensor product of an arbitrary number of vector spaces.

A. Decomposition into direct sum of irreducible representations

Let $\mathbb{V}$ be a finite-dimensional space and let $\varphi \in [0,2\pi)$ label a set of linear transformations $\hat{W}_\varphi$,

$$\hat{W}_\varphi : \mathbb{V} \rightarrow \mathbb{V}, \tag{17}$$

that are a unitary representation of the group U(1). That is,

$$\hat{W}_\varphi^\dagger \hat{W}_\varphi = \hat{W}_\varphi \hat{W}_\varphi^\dagger = \mathbb{I}, \quad \forall \, \varphi \in [0,2\pi), \tag{18}$$

$$\hat{W}_{\varphi_1} \hat{W}_{\varphi_2} = \hat{W}_{\varphi_2} \hat{W}_{\varphi_1} = \hat{W}_{\varphi_1 + \varphi_2|_{2\pi}}, \quad \forall \, \varphi_1, \varphi_2 \in [0,2\pi). \tag{19}$$

Then $\mathbb{V}$ decomposes as the direct sum of (possibly degenerate) one-dimensional irreducible representations (or *irreps*) of U(1),

$$\mathbb{V} \cong \bigoplus_n \mathbb{V}_n, \tag{20}$$

where $\mathbb{V}_n$ is a subspace of dimension d_n, made of d_n copies of an irrep of U(1) with charge $n \in \mathbb{Z}$. We say that irrep n is d_n-fold degenerate and that $\mathbb{V}_n$ is the degeneracy space. For concreteness, in this paper we identify the integer charge n as labeling the number of particles (another frequent identification is with the z component of the spin, in which case semi-integer numbers may be considered). The representation of group U(1) is generated by the particle number operator $\hat{n}$,

$$\hat{n} \equiv \sum_n n \hat{P}_n, \quad \hat{P}_n \equiv \sum_{t_n=1}^{d_n} |nt_n\rangle\langle nt_n|, \tag{21}$$

where $\hat{P}_n$ is a projector onto the subspace $\mathbb{V}_n$ of particle number n, and the vectors $|nt_n\rangle$,

$$\hat{n}|nt_n\rangle = n|nt_n\rangle, \quad t_n = 1,\ldots,d_n, \tag{22}$$

are an orthonormal basis of $\mathbb{V}_n$. In terms of $\hat{n}$, the transformations $\hat{W}_\varphi$ read

$$\hat{W}_\varphi = e^{-i\hat{n}\varphi}. \tag{23}$$

It then follows from Eq. (22) that

$$\hat{W}_\varphi|nt_n\rangle = e^{-in\varphi}|nt_n\rangle, \quad \forall \, \varphi \in [0,2\pi). \tag{24}$$

The dual basis $\{\langle nt_n|\}$ is transformed by the *dual representation* of U(1), with elements $\hat{W}_\varphi^\dagger$, as

$$\langle nt_n|\hat{W}_\varphi^\dagger = e^{in\varphi}\langle nt_n|, \quad \forall \, \varphi \in [0,2\pi). \tag{25}$$

Example 1. Consider a two-dimensional space $\mathbb{V}$ that decomposes as $\mathbb{V} \cong \mathbb{V}_0 \oplus \mathbb{V}_1$, where the irreps $n = 0$ and $n = 1$ are nondegenerate (i.e., $d_0 = d_1 = 1$). Then the orthogonal vectors $\{|n = 0,t_0 = 1\rangle, |n = 1,t_1 = 1\rangle\}$ form a basis of $\mathbb{V}$. In column vector notation,

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \equiv |n = 0,t_0 = 1\rangle, \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix} \equiv |n = 1,t_1 = 1\rangle, \tag{26}$$

the particle number operator $\hat{n}$ and transformation $\hat{W}_\varphi$ read

$$\hat{n} \equiv \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \quad \hat{W}_\varphi \equiv \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\varphi} \end{pmatrix}. \tag{27}$$

Example 2. Consider a four-dimensional space $\mathbb{V}$ that decomposes as $\mathbb{V} \cong \mathbb{V}_0 \oplus \mathbb{V}_1 \oplus \mathbb{V}_2$, where $d_0 = d_2 = 1$ and $d_1 = 2$, so that now irrep $n = 1$ is twofold degenerate. Let $\{|n = 1,t_1 = 1\rangle, |n = 1,t_1 = 2\rangle\}$ form a basis of $\mathbb{V}_1$. In column vector notation,

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \equiv |n = 0,t_0 = 1\rangle, \quad \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \equiv |n = 1,t_1 = 1\rangle, \tag{28}$$

$$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \equiv |n = 1,t_1 = 2\rangle, \quad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \equiv |n = 2,t_2 = 1\rangle, \tag{29}$$

the particle number operator $\hat{n}$ and transformation $\hat{W}_\varphi$ read

$$\hat{n} \equiv \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}, \quad \hat{W} \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{-i\varphi} & 0 & 0 \\ 0 & 0 & e^{-i\varphi} & 0 \\ 0 & 0 & 0 & e^{-i2\varphi} \end{pmatrix}. \tag{30}$$

B. Symmetric states and operators

In this work we are interested in states and operators that have a simple transformation rule under the action of U(1). A pure state $|\Psi\rangle \in \mathbb{V}$ is *symmetric* if it transforms as

$$\hat{W}_\varphi|\Psi\rangle = e^{-in\varphi}|\Psi\rangle, \quad \forall \, \varphi \in [0,2\pi). \tag{31}$$

The case $n = 0$ corresponds to an *invariant* state, $\hat{W}_\varphi|\Psi\rangle = |\Psi\rangle$, which transforms trivially under U(1), whereas for $n \neq 0$ the state is *covariant*, with $|\Psi\rangle$ being multiplied by a nontrivial phase $e^{-in\varphi}$. Notice that a symmetric state $|\Psi\rangle$ is an eigenstate of $\hat{n}$: that is, it has a well-defined particle number n. $|\Psi\rangle$ can

thus be expanded in terms of a basis of the relevant subspace $\mathbb{V}_n$,

$$\hat{n}|\Psi\rangle = n|\Psi\rangle, \quad |\Psi\rangle = \sum_{t_n=1}^{d_n}(\hat{\Psi}_n)_{t_n}|nt_n\rangle, \quad (32)$$

where we have introduced a charge label $_n$ on the state coefficients of $|\Psi\rangle$ so that we can explicitly associate each coefficient $(\hat{\Psi}_n)_{t_n}$ with its corresponding basis vector $|nt_n\rangle$.

A linear operator $\hat{T} : \mathbb{V} \to \mathbb{V}$ is invariant if it commutes with the generator $\hat{n}$,

$$[\hat{T}, \hat{n}] = 0, \quad (33)$$

or equivalently if it commutes with the action of the group,

$$\hat{W}_\varphi \hat{T} \hat{W}_\varphi^\dagger = \hat{T}, \quad \forall \varphi \in [0, 2\pi). \quad (34)$$

It follows that $\hat{T}$ decomposes as (Schur's lemma)

$$\hat{T} = \bigoplus_n \hat{T}_n, \quad (35)$$

where $\hat{T}_n$ is a $d_n \times d_n$ matrix that acts on the subspace $\mathbb{V}_n$ in Eq. (20).

Notice that the operator $\hat{T}$ in Eq. (35) transforms vectors with a well-defined particle number n into vectors with the same particle number. That is, U(1)-invariant operators *conserve particle number*.

Example 1 revisited. In example 1 above, symmetric vectors must be proportional to either $|n = 0, t_0 = 1\rangle$ or $|n = 1, t_1 = 1\rangle$. An invariant operator $\hat{T} = \hat{T}_0 \oplus \hat{T}_1$ is of the form

$$\hat{T} = \begin{pmatrix} \alpha_0 & 0 \\ 0 & \alpha_1 \end{pmatrix}, \quad \alpha_0, \alpha_1 \in \mathbb{C}. \quad (36)$$

Example 2 revisited. In example 2 above, a symmetric vector $|\Psi\rangle$ must be of the form

$$|\Psi\rangle = \begin{pmatrix} \alpha_0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |\Psi\rangle = \begin{pmatrix} 0 \\ \alpha_1 \\ \beta_1 \\ 0 \end{pmatrix}, \quad \text{or} \quad |\Psi\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \alpha_2 \end{pmatrix}, \quad (37)$$

where $\alpha_0, \alpha_1, \beta_1, \alpha_2 \in \mathbb{C}$. An invariant operator $\hat{T} = \hat{T}_0 \oplus \hat{T}_1 \oplus \hat{T}_2$ is of the form

$$\hat{T} = \begin{pmatrix} \alpha_0 & 0 & 0 & 0 \\ 0 & \alpha_1 & \beta_1 & 0 \\ 0 & \gamma_1 & \delta_1 & 0 \\ 0 & 0 & 0 & \alpha_2 \end{pmatrix}, \quad (38)$$

where $\hat{T}_1$ corresponds to the 2×2 central block and $\alpha_0, \alpha_1, \beta_1, \gamma_1, \delta_1, \alpha_2 \in \mathbb{C}$.

The above examples illustrate that the symmetry imposes constraints on vectors and operators. By using an eigenbasis $\{|nt_n\rangle\}$ of the particle number operator $\hat{n}$, these constraints imply the presence of the zeros in Eqs. (36)–(38). Thus a reduced number of complex coefficients is required in order to describe U(1)-symmetric vectors and operators. As we will discuss in Sec. IV, performing manipulations on

symmetric tensors can also result in a significant reduction in computational costs.

C. Tensor product of two representations

Let $\mathbb{V}^{(A)}$ and $\mathbb{V}^{(B)}$ be two spaces that carry representations of U(1), as generated by particle number operators $\hat{n}^{(A)}$ and $\hat{n}^{(B)}$, and let

$$\mathbb{V}^{(A)} \cong \bigoplus_{n_A} \mathbb{V}_{n_A}^{(A)}, \quad \mathbb{V}^{(B)} \cong \bigoplus_{n_B} \mathbb{V}_{n_B}^{(B)} \quad (39)$$

be their decompositions as a direct sum of (possibly degenerate) irreps. Let us also consider the action of U(1) on the tensor product $\mathbb{V}^{(AB)} \cong \mathbb{V}^{(A)} \otimes \mathbb{V}^{(B)}$ as generated by the *total particle number operator*

$$\hat{n}^{(AB)} \equiv \hat{n}^{(A)} \otimes \mathbb{I} + \mathbb{I} \otimes \hat{n}^{(B)}, \quad (40)$$

that is, implemented by unitary transformations

$$\hat{W}_\varphi^{(AB)} \equiv e^{-i\hat{n}^{(AB)}\varphi}. \quad (41)$$

The space $\mathbb{V}^{(AB)}$ also decomposes as the direct sum of (possibly degenerate) irreps,

$$\mathbb{V}^{(AB)} \cong \bigoplus_{n_{AB}} \mathbb{V}_{n_{AB}}^{(AB)}. \quad (42)$$

Here the subspace $\mathbb{V}_{n_{AB}}^{(AB)}$, with total particle number n_{AB}, corresponds to the direct sum of all products of subspaces $\mathbb{V}_{n_A}^{(A)}$ and $\mathbb{V}_{n_B}^{(B)}$ such that $n_A + n_B = n_{AB}$,

$$\mathbb{V}_{n_{AB}}^{(AB)} \cong \bigoplus_{n_A, n_B | n_A + n_B = n_{AB}} \mathbb{V}_{n_A}^{(A)} \otimes \mathbb{V}_{n_B}^{(B)}. \quad (43)$$

For each subspace $\mathbb{V}_{n_{AB}}^{(AB)}$ in Eq. (42) we introduce a *coupled basis* $\{|n_{AB}t_{n_{AB}}\rangle\}$,

$$\hat{n}^{(AB)}|n_{AB}t_{n_{AB}}\rangle = n_{AB}|n_{AB}t_{n_{AB}}\rangle, \quad (44)$$

where each vector $|n_{AB}t_{n_{AB}}\rangle$ corresponds to the tensor product $|n_A t_{n_A}; n_B t_{n_B}\rangle \equiv |n_A t_{n_A}\rangle \otimes |n_B t_{n_B}\rangle$ of a unique pair of vectors $|n_A t_{n_A}\rangle$ and $|n_B t_{n_B}\rangle$, with $n_A + n_B = n_{AB}$. Let table Υ^{fuse}, with components

$$\Upsilon_{n_A t_{n_A}, n_B t_{n_B} \to n_{AB} t_{n_{AB}}}^{\text{fuse}} \equiv \langle n_{AB}t_{n_{AB}}|n_A t_{n_A}; n_B t_{n_B}\rangle, \quad (45)$$

encode this one-to-one correspondence. Notice that each component of Υ^{fuse} is either a 0 or a 1. Then

$$|n_{AB}t_{n_{AB}}\rangle = \sum_{n_A t_{n_A} n_B t_{n_B}} \Upsilon_{n_A t_{n_A}, n_B t_{n_B} \to n_{AB} t_{n_{AB}}}^{\text{fuse}}$$
$$\times |n_A t_{n_A}; n_B t_{n_B}\rangle. \quad (46)$$

For later reference (see the Appendix), we notice that Υ^{fuse} can be decomposed into two pieces. The first piece expresses a basis $\{|n_A t_{n_A}; n_B t_{n_B}\rangle\}$ of $\mathbb{V}^{(AB)}$ in terms of the basis $\{|n_A t_{n_A}\rangle\}$ of $\mathbb{V}^{(A)}$ and the basis $\{|n_B t_{n_B}\rangle\}$ of $\mathbb{V}^{(B)}$. This assignment occurs as in the absence of the symmetry, where one creates a composed index $d = b \times c$ by running, for example, fast over index c and slowly over index b as in Eq. (2). Note that this procedure does not always lead to the set $\{|n_A t_{n_A}; n_B t_{n_B}\rangle\}$ being ordered such that states corresponding to the same total particle number $n_{AB} = n_A + n_B$ are adjacent to each other within the set. This ordering is achieved by the second piece: a permutation of

basis elements that reorganizes them according to their total particle number n_{AB}, so that they are identified in a one-to-one correspondence with the coupled states $\{|n_{AB}t_{n_{AB}}\rangle\}$.

Finally, the product basis can be expressed in terms of the coupled basis

$$|n_A t_{n_A}; n_B t_{n_B}\rangle = \sum_{n_{AB}t_{n_{AB}}} \Upsilon^{\text{split}}_{n_{AB}t_{n_{AB}} \rightarrow n_A t_{n_A}, n_B t_{n_B}}$$
$$\times |n_{AB}t_{n_{AB}}\rangle, \quad (47)$$

with

$$\Upsilon^{\text{split}}_{n_{AB}t_{n_{AB}} \rightarrow n_A t_{n_A}, n_B t_{n_B}} = \Upsilon^{\text{fuse}}_{n_A t_{n_A}, n_B t_{n_B} \rightarrow n_{AB}t_{n_{AB}}}. \quad (48)$$

Example 3. Consider the case where both $\mathbb{V}^{(A)}$ and $\mathbb{V}^{(B)}$ correspond to the space of example 1, that is, $\mathbb{V}^{(A)} \cong \mathbb{V}_0^{(A)} \oplus \mathbb{V}_1^{(A)}$ and $\mathbb{V}^{(B)} \cong \mathbb{V}_0^{(B)} \oplus \mathbb{V}_1^{(B)}$, where $\mathbb{V}_0^{(A)}$, $\mathbb{V}_1^{(A)}$, $\mathbb{V}_0^{(B)}$, and $\mathbb{V}_1^{(B)}$ all have dimension 1. Then $\mathbb{V}^{(AB)}$ corresponds to the space in example 2, namely

$$\begin{aligned} \mathbb{V}^{(AB)} &\cong \mathbb{V}^{(A)} \otimes \mathbb{V}^{(B)} \\ &\cong \left(\mathbb{V}_0^{(A)} \oplus \mathbb{V}_1^{(A)}\right) \otimes \left(\mathbb{V}_0^{(B)} \oplus \mathbb{V}_1^{(B)}\right) \\ &\cong \mathbb{V}_0^{(AB)} \oplus \mathbb{V}_1^{(AB)} \oplus \mathbb{V}_2^{(AB)}, \end{aligned} \quad (49)$$

where

$$\mathbb{V}_0^{(AB)} \cong \mathbb{V}_0^{(A)} \otimes \mathbb{V}_0^{(B)}, \quad (50)$$

$$\mathbb{V}_1^{(AB)} \cong \left(\mathbb{V}_0^{(A)} \otimes \mathbb{V}_1^{(B)}\right) \oplus \left(\mathbb{V}_1^{(A)} \otimes \mathbb{V}_0^{(B)}\right), \quad (51)$$

$$\mathbb{V}_2^{(AB)} \cong \mathbb{V}_1^{(A)} \otimes \mathbb{V}_1^{(B)}. \quad (52)$$

The coupled basis $\{|n_{AB}t_{n_{AB}}\rangle\}$ reads

$$\begin{aligned} |n_{AB} = 0, t_0 = 1\rangle &= |n_A = 0, t_0 = 1\rangle \otimes |n_B = 0, t_0 = 1\rangle, \\ |n_{AB} = 1, t_1 = 1\rangle &= |n_A = 0, t_0 = 1\rangle \otimes |n_B = 1, t_1 = 1\rangle, \\ |n_{AB} = 1, t_1 = 2\rangle &= |n_A = 1, t_1 = 1\rangle \otimes |n_B = 0, t_0 = 1\rangle, \\ |n_{AB} = 2, t_2 = 1\rangle &= |n_A = 1, t_1 = 1\rangle \otimes |n_B = 1, t_1 = 1\rangle, \end{aligned}$$
$$(53)$$

where we emphasize that the degeneracy index $t_{n_{AB}}$ takes two possible values for $n_{AB} = 1$, i.e., $t_1 \in \{1,2\}$, since there are two states $|n_A t_{n_A}\rangle \otimes |n_B t_{n_B}\rangle$ with $n_A + n_B = 1$. The components $\Upsilon^{\text{fuse}}_{n_A t_A, n_B t_B \rightarrow n_{AB}t_{AB}}$ of the tensor Υ^{fuse} that encodes this change of basis are all zero except for

$$\Upsilon^{\text{fuse}}_{01,01 \rightarrow 01} = \Upsilon^{\text{fuse}}_{01,11 \rightarrow 11} = \Upsilon^{\text{fuse}}_{11,01 \rightarrow 12} = \Upsilon^{\text{fuse}}_{11,11 \rightarrow 21} = 1.$$

D. Lattice models with U(1) symmetry

The action of U(1) on the threefold tensor product,

$$\mathbb{V}^{(ABC)} \cong \mathbb{V}^{(A)} \otimes \mathbb{V}^{(B)} \otimes \mathbb{V}^{(C)}, \quad (54)$$

as generated by the total particle number operator,

$$\hat{n}^{(ABC)} = \hat{n}^{(A)} \otimes \mathbb{I} \otimes \mathbb{I} + \mathbb{I} \otimes \hat{n}^{(B)} \otimes \mathbb{I} + \mathbb{I} \otimes \mathbb{I} \otimes \hat{n}^{(C)},$$
$$(55)$$

induces a decomposition

$$\mathbb{V}^{(ABC)} \cong \bigoplus_{n_{ABC}} \mathbb{V}_{n_{ABC}}^{(ABC)} \quad (56)$$

in terms of irreps $\mathbb{V}_{n_{ABC}}^{(ABC)}$ which we can now relate to $\mathbb{V}_{n_A}^{(A)}$, $\mathbb{V}_{n_B}^{(B)}$, and $\mathbb{V}_{n_C}^{(C)}$. For example, we can consider first the product $\mathbb{V}_{n_{AB}}^{(AB)} \cong \mathbb{V}_{n_A}^{(A)} \otimes \mathbb{V}_{n_B}^{(B)}$ and then the product $\mathbb{V}_{n_{ABC}}^{(ABC)} \cong \mathbb{V}_{n_{AB}}^{(AB)} \otimes \mathbb{V}_{n_C}^{(C)}$, using a different table Υ^{fuse} at each step to relate the coupled basis to the product basis as discussed in the previous section. Similarly we could consider the action of U(1) on four tensor products, and so on.

In particular we will be interested in a lattice $\mathcal{L}$ made of L sites with vector space $\mathbb{V}^{\otimes L}$, where for simplicity we will assume that each site $s \in \mathcal{L}$ is described by the same finite-dimensional vector space $\mathbb{V}$ (see Sec. II E). Given a particle number operator $\hat{n}$ defined on each site, we can consider the action of U(1) generated by the total particle number operator

$$\hat{N} \equiv \sum_{s=1}^{L} \hat{n}^{(s)}, \quad (57)$$

which corresponds to unitary transformations

$$W_\varphi^{[L]} \equiv e^{-i\hat{N}\varphi} = (e^{-i\hat{n}\varphi})^{\otimes L} = (\hat{W}_\varphi)^{\otimes L}. \quad (58)$$

The tensor product space $\mathbb{V}^{\otimes L}$ decomposes as

$$\mathbb{V}^{\otimes L} \cong \bigoplus_{N} \mathbb{V}_N \quad (59)$$

and we denote by $\{|Nt_N\rangle\}$ the particle number basis in $\mathbb{V}^{\otimes L}$.

We say that a lattice model is *U(1) symmetric* if its Hamiltonian $\hat{H} : \mathbb{V} \rightarrow \mathbb{V}$ commutes with the action of the group. That is,

$$[\hat{H}, \hat{N}] = 0, \quad (60)$$

or equivalently

$$(\hat{W}_\varphi)^{\otimes L} \hat{H} (\hat{W}_\varphi^\dagger)^{\otimes L} = \hat{H}, \quad \forall \varphi \in [0, 2\pi). \quad (61)$$

One example of a U(1)-symmetric model is the hard-core Bose-Hubbard model, with Hamiltonian

$$\hat{H}_{\text{HCBH}} \equiv \sum_{s=1}^{L} (\hat{a}_s^\dagger \hat{a}_{s+1} + \hat{a}_s \hat{a}_{s+1}^\dagger + \gamma \hat{n}_s \hat{n}_{s+1}) - \mu \sum_{s=1}^{L} \hat{n}_s,$$
$$(62)$$

where we consider periodic boundary conditions (by identifying sites $L + 1$ and 1), and $\hat{a}_s^\dagger$ and $\hat{a}_s$ are hard-core bosonic creation and annihilation operators, respectively. In terms of the basis introduced in example 1 these operators are defined as

$$\hat{a} \equiv \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad \hat{n} \equiv \hat{a}^\dagger \hat{a} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

To see that $\hat{H}_{\text{HCBH}}$ commutes with the action of the group, we first observe that for two sites

$$[\hat{a}_1^\dagger \hat{a}_2 + \hat{a}_2^\dagger \hat{a}_1, \hat{n}_1 + \hat{n}_2] = 0, \quad (63)$$

from which it readily follows that $[\hat{H}_{\text{HCBH}}, \hat{N}] = 0$.

Notice that the chemical potential term $-\mu \sum_s \hat{n}_s = -\mu \hat{N}$ also commutes with the rest of the Hamiltonian. The ground state $|\Psi_N^{GS}\rangle$ of $\hat{H}_{HCBH}$ in a particular subspace $\mathbb{V}_N$ or particle number sector can be turned into the absolute ground state by tuning the chemical potential μ. This fact can be used to find the ground state $|\Psi_N^{GS}\rangle$ of any particle number sector through an algorithm which can only minimize the expectation value of $\hat{H}_{HCBH}$. However, we will later see that the use of symmetric tensors in the context of tensor network states will allow us to directly minimize the expectation value of $\hat{H}_{HCBH}$ in a given particle number sector by restricting the search to states

$$|\Psi_N\rangle = \sum_{t_N=1}^{d_N} (\hat{\Psi}_N)_{t_N} |N t_N\rangle \qquad (64)$$

with the desired particle number N.

Finally, by making the identifications

$$\hat{n} = \frac{\mathbb{I} - \hat{\sigma}_z}{2}, \quad \hat{a} = \frac{\hat{\sigma}_x + i\hat{\sigma}_y}{2},$$

where $\hat{\sigma}_x, \hat{\sigma}_y, \hat{\sigma}_z$ are the Pauli matrices

$$\hat{\sigma}_x \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \hat{\sigma}_y \equiv \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \hat{\sigma}_z \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \qquad (65)$$

one can map $\hat{H}_{HCBH}$ to the spin-$\frac{1}{2}$ XXZ quantum spin chain

$$\hat{H}_{XXZ} \equiv \sum_{s=1}^{L} \left(\hat{\sigma}_x^{(s)} \hat{\sigma}_x^{(s+1)} + \hat{\sigma}_y^{(s)} \hat{\sigma}_y^{(s+1)} + \Delta \hat{\sigma}_z^{(s)} \hat{\sigma}_z^{(s+1)} \right), \quad (66)$$

where we have ignored terms proportional to $\hat{N}$ and set $\Delta \equiv \gamma/4$. In particular, for $\Delta = 0$ we obtain the quantum XX spin chain

$$\hat{H}_{XX} \equiv \sum_{s=1}^{L} \left(\hat{\sigma}_x^{(s)} \hat{\sigma}_x^{(s+1)} + \hat{\sigma}_y^{(s)} \hat{\sigma}_y^{(s+1)} \right), \qquad (67)$$

and for $\gamma = 1$, the quantum Heisenberg spin chain

$$\hat{H}_{XXX} \equiv \sum_{s=1}^{L} \left(\hat{\sigma}_x^{(s)} \hat{\sigma}_x^{(s+1)} + \hat{\sigma}_y^{(s)} \hat{\sigma}_y^{(s+1)} + \hat{\sigma}_z^{(s)} \hat{\sigma}_z^{(s+1)} \right). \quad (68)$$

In Sec. V, the quantum spin models (67) and (68) will be used to benchmark the performance increase resulting from the use of symmetries in tensor networks algorithms.

IV. TENSOR NETWORKS WITH U(1) SYMMETRY

In this section we consider U(1)-symmetric tensors and tensor networks. We explain how to decompose U(1)-symmetric tensors in a compact, canonical form that exploits their symmetry. We then discuss how to adapt the set $\mathcal{P}$ of primitives for tensor network manipulations in order to work in this form. We also analyze how working in the canonical form affects computational costs.

A. U(1)-symmetric tensors

Let $\hat{T}$ be a rank-k tensor with components $\hat{T}_{i_1 i_2 \cdots i_k}$. As in Sec. II F, we regard tensor $\hat{T}$ as a linear map between the vector spaces $\mathbb{V}^{[in]}$ and $\mathbb{V}^{[out]}$ [Eq. (15)]. This implies that

each index is either an incoming or outgoing index. On each space $\mathbb{V}^{[i_l]}$, associated with index i_l, we introduce a particle number operator $\hat{n}^{(l)}$ that generates a unitary representation of U(1) given by matrices $\hat{W}_\varphi^{(l)} \equiv e^{-i\hat{n}^{(l)}\varphi}$, $\varphi \in [0, 2\pi)$. In the following, we use $\hat{W}_\varphi^{(l)*}$ to denote the complex conjugate of $\hat{W}_\varphi^{(l)}$.

Let us consider the action of U(1) on the space

$$\mathbb{V}^{[i_1]} \otimes \mathbb{V}^{[i_2]} \otimes \cdots \otimes \mathbb{V}^{[i_k]} \qquad (69)$$

given by

$$\hat{X}_\varphi^{(1)} \otimes \hat{X}_\varphi^{(2)} \otimes \cdots \otimes \hat{X}_\varphi^{(k)}, \qquad (70)$$

where

$$\hat{X}_\varphi^{(l)} = \left\{ \begin{array}{ll} \hat{W}_\varphi^{(l)*} & \text{if} \quad i_l \in I, \\ \hat{W}_\varphi^{(l)} & \text{if} \quad i_l \in O. \end{array} \right\} \qquad (71)$$

That is, $\hat{X}_\varphi^{(l)}$ acts differently depending on whether index i_l of tensor $\hat{T}$ is an incoming or outgoing index. We then say that tensor $\hat{T}$, with components $T_{i_1 i_2 \cdots i_k}$, is *U(1) invariant* if it is invariant under the transformation of Eq. (70),

$$\sum_{i_1, i_2, \ldots, i_k} \left(\hat{X}_\varphi^{(1)} \right)_{i_1' i_1} \left(\hat{X}_\varphi^{(2)} \right)_{i_2' i_2} \cdots \left(\hat{X}_\varphi^{(k)} \right)_{i_k' i_k} \hat{T}_{i_1 i_2 \cdots i_k} = \hat{T}_{i_1' i_2' \cdots i_k'}, \quad (72)$$

for all $\varphi \in [0, 2\pi)$. This is depicted in Fig. 9.

Example 4. A U(1)-invariant vector $|\Psi\rangle$—that is, a vector with $\hat{n}|\Psi\rangle = 0$ and components $(\hat{\Psi}_{n=0})_{t_0}$ in the subspace $\mathbb{V}_{n=0}$ which corresponds to vanishing particle number $n = 0$ [cf. Eq. (32)]—fulfills

$$(\hat{\Psi}_{n=0})_{t_{0'}} = \sum_{t_0} (\hat{W}_\varphi)_{t_{0'} t_0} (\hat{\Psi}_{n=0})_{t_0}, \quad \forall \varphi \in [0, 2\pi), \quad (73)$$

in accordance with Eq. (31), as shown in Fig. 9(a).

Example 5. A U(1)-invariant matrix $\hat{T}$ (35) fulfills

$$\hat{T}_{a'b'} = \sum_{a,b} (\hat{W}_\varphi)_{a'a} (\hat{W}_\varphi^*)_{b'b} \hat{T}_{ab} \qquad (74)$$

$$= \sum_{a,b} (\hat{W}_\varphi)_{a'a} \hat{T}_{ab} (\hat{W}_\varphi^\dagger)_{bb'}, \quad \forall \varphi \in [0, 2\pi), \quad (75)$$

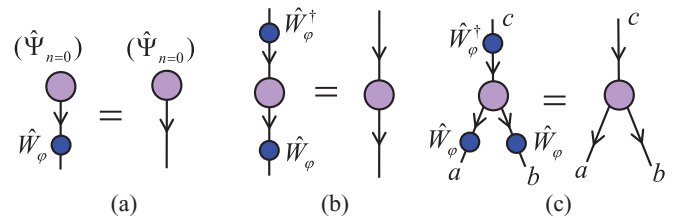in accordance with Eq. (34) [see Fig. 9(b)].

FIG. 9. (Color online) (a) Constraint fulfilled by a U(1)-invariant vector. The only allowed particle number on the single index is $n = 0$. (b) Constraint fulfilled by a U(1)-invariant matrix. It follows from Schur's lemma that the matrix is block diagonal in particle number. (c) Constraint fulfilled by a rank-3 tensor with one incoming index and two outgoing indices.

Example 6. Tensor $\hat{T}$ in Eq. (16), with components $\hat{T}_{abc}$, where a and b are outgoing indices and c is an incoming index, is U(1) invariant if

$$\hat{T}_{a'b'c'} = \sum_{a,b,c} \left(\hat{W}_\varphi^{(1)}\right)_{a'a} \left(\hat{W}_\varphi^{(2)}\right)_{b'b} \left(\hat{W}_\varphi^{(3)*}\right)_{c'c} \hat{T}_{abc} \quad (76)$$

$$= \sum_{a,b,c} \left(\hat{W}_\varphi^{(1)}\right)_{a'a} \left(\hat{W}_\varphi^{(2)}\right)_{b'b} \hat{T}_{abc} \left(\hat{W}_\varphi^{(3)\dagger}\right)_{cc'} \quad (77)$$

for all $\varphi \in [0,2\pi)$ [see Fig. 9(c)].

Further, we say that a tensor $\hat{Q}$ with components $\hat{Q}_{i_1 i_2 \cdots i_k}$ is *U(1) covariant* if under the transformation of Eq. (70) it acquires a non trivial phase $e^{-in\varphi}$,

$$\sum_{i_1,i_2,\ldots,i_k} \left(\hat{X}_\varphi^{(1)}\right)_{i_1' i_1} \left(\hat{X}_\varphi^{(2)}\right)_{i_2 i_2'} \cdots \left(\hat{X}_\varphi^{(k)}\right)_{i_k' i_k} \hat{Q}_{i_1 i_2 \cdots i_k} = e^{-in\varphi} \hat{Q}_{i_1' i_2' \cdots i_k'},$$
$$(78)$$

for all $\varphi \in [0,2\pi)$.

Example 7. A U(1)-covariant vector $|\Psi\rangle$—that is, one which satisfies $\hat{n}|\Psi\rangle = n|\Psi\rangle$ for some $n \neq 0$, and has nonzero components $(\hat{\Psi}_n)_{t_n}$ only in the relevant subspace $\mathbb{V}_n$ [cf. Eq. (32)]—fulfills

$$\sum_{t_n} (\hat{W}_\varphi)_{t_n' t_n} (\hat{\Psi}_n)_{t_n} = e^{-in\varphi} (\hat{\Psi}_n)_{t_n'}, \quad \forall \varphi \in [0,2\pi), \quad (79)$$

in accordance with Eq. (31). (See also Fig. 10.)

Notice that we can describe the rank-k covariant tensor $\hat{Q}$ above by a rank-$(k+1)$ invariant tensor $\hat{T}$ with components

$$\hat{T}_{i_1 i_2 \cdots i_k i} \equiv \hat{Q}_{i_1 i_2 \cdots i_k}, \qquad |i| = 1. \quad (80)$$

This is built from $\hat{Q}$ by adding an extra incoming index i, where the index i has a fixed particle number n and no degeneracy (i.e., i is associated to a trivial space $\mathbb{V}^{[i]} \cong \mathbb{C}$). We refer to both *invariant* and *covariant* tensors as *symmetric* tensors. By using the above construction, in this work we will represent all U(1)-symmetric tensors by means of U(1)-invariant tensors. In particular, we represent the nontrivial components $(\hat{\Psi}_n)_{t_n}$ of the covariant vector $|\Psi_n\rangle$ in Eqs. (31) and (32) as an invariant matrix $\hat{T}$ of size $|t_n| \times 1$ with components $\hat{T}_{t_n 1} = (\hat{\Psi}_n)_{t_n}$. Consequently, from now on we will mostly consider only invariant tensors.
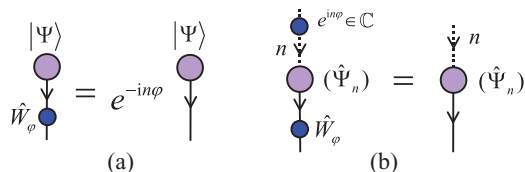

FIG. 10. (Color online) (a) U(1)-covariant vector $|\Psi\rangle$, with some nonvanishing particle number $n \neq 0$. Under the action of U(1) on its index, the covariant vector $|\Psi\rangle$ acquires a phase $e^{-in\varphi}$ [Eq. (79)]. (b) The U(1)-covariant vector $|\Psi\rangle$, with components $(\Psi_n)_{t_n}$, can be represented by a U(1)-invariant matrix $\hat{T}$ with components $\hat{T}_{t_1 i} = (\Psi_n)_{i_1}$, where i is a trivial index ($|i| = 1$) with charge n and is decorated by the opposite arrow to i_1.

B. Canonical form for U(1)-invariant tensors

Let us now write a tensor $\hat{T}$ in a particle number basis on each factor space in Eq. (69). That is, each index $i_1, i_2, \ldots, i_k$ is decomposed into a particle number index n and a degeneracy index t_n, $i_1 = (n_1, t_{n_1}), i_2 = (n_2, t_{n_2}), \ldots, i_k = (n_k, t_{n_k})$, and

$$\hat{T}_{i_1 i_2 \cdots i_k} \equiv \left(\hat{T}_{n_1 n_2 \cdots n_k}\right)_{t_{n_1} t_{n_2} \cdots t_{n_k}}. \quad (81)$$

Here, for each set of particle numbers $n_1, n_2, \ldots, n_k$, we regard $\hat{T}_{n_1 n_2 \cdots n_k}$ as a tensor with components $(\hat{T}_{n_1 n_2 \cdots n_k})_{t_{n_1} t_{n_2} \cdots t_{n_k}}$. Let N_{in} and N_{out} denote the sum of particle numbers corresponding to incoming and outgoing indices,

$$N_{\text{in}} \equiv \sum_{n_l \in I} n_l, \quad N_{\text{out}} \equiv \sum_{n_l \in O} n_l. \quad (82)$$

The condition for a nonvanishing tensor of the form $\hat{T}_{n_1 n_2 \cdots n_k}$ to be invariant under U(1), Eq. (70), is simply that the sum of incoming particle numbers equals the sum of outgoing particle numbers. Therefore a U(1)-invariant tensor $\hat{T}$ satisfies

$$\hat{T} = \bigoplus_{n_1, n_2, \ldots, n_k} \hat{T}_{n_1 n_2 \cdots n_k} \delta_{N_{\text{in}}, N_{\text{out}}}. \quad (83)$$

[We use the direct sum symbol $\bigoplus$ to denote that the different tensors $\hat{T}_{n_1 n_2 \cdots n_k}$ are supported on orthonormal subspaces of the tensor product space of Eq. (69).] In components, the above expression reads

$$\hat{T}_{i_1 i_2 \cdots i_k} \equiv \left(\hat{T}_{n_1 n_2 \cdots n_k}\right)_{t_{n_1} t_{n_2} \cdots t_{n_k}} \delta_{N_{\text{in}}, N_{\text{out}}}. \quad (84)$$

Here, $\delta_{N_{\text{in}}, N_{\text{out}}}$ implements particle number conservation: if $N_{\text{in}} \neq N_{\text{out}}$, then all components of $\hat{T}_{n_1 n_2 \cdots n_k}$ must vanish. This generalizes the block structure of U(1)-invariant matrices in Eq. (35) (where $\hat{T}_{nn}$ is denoted $\hat{T}_n$) to tensors of arbitrary rank k. The canonical decomposition in Eq. (83) is important, in that it allows us to identify the degrees of freedom of tensor $\hat{T}$ that are not determined by the symmetry. Expressing tensor $\hat{T}$ in terms of the tensors $\hat{T}_{n_1 n_2 \cdots n_k}$ with $N_{\text{in}} = N_{\text{out}}$ ensures that we store $\hat{T}$ in the most compact way possible.

Notice that the canonical form of Eq. (83) is a particular case of the canonical form presented in Eq. (15) of Ref. 52 for more general (possibly non-Abelian) symmetry groups. There, a symmetric tensor was decomposed into *degeneracy* tensors [analogous to tensors $\hat{T}_{n_1 n_2 \cdots n_k}$ in Eq. (83)] and *structural* tensors [generalizing the term $\delta_{N_{\text{in}}, N_{\text{out}}}$ in Eq. (83)] which can in general be expanded as a trivalent network of Clebsch-Gordan (or coupling) coefficients of the symmetry group. In the case of non-Abelian groups, where some irreps have dimension larger than 1, the structural tensors are highly nontrivial. However, for the group U(1) discussed in this paper (as for any other Abelian group) all irreps are one dimensional and the structural tensors are always reduced to a simple expression such as $\delta_{N_{\text{in}}, N_{\text{out}}}$ in Eq. (83). (Nevertheless, in the Appendix we will resort to a more elaborate decomposition of the structural tensors in order to better exploit the presence of symmetry in those tensor network algorithms based on iterating a fixed sequence of manipulations.)

C. U(1)-symmetric tensor networks

In Sec. II F we saw that a tensor network $\mathcal{N}$ where each line has a direction (represented with an arrow) can be interpreted
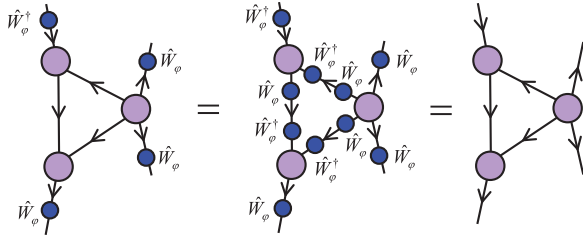
FIG. 11. (Color online) A tensor network $\mathcal{N}$ made of U(1)-invariant tensors represents a U(1)-invariant tensor $\hat{T}$. This is seen by means of two equalities. The first equality is obtained by inserting resolutions of the identity $\mathbb{I} = \hat{W}_\varphi \hat{W}_\varphi^\dagger$ on each index connecting two tensors in $\mathcal{N}$. The second equality follows from the fact that each tensor in $\mathcal{N}$ is U(1) invariant.

as a collection of linear maps composed into a single linear map $\hat{T}$ of which $\mathcal{N}$ is a tensor network decomposition. By introducing a particle number operator on the vector space associated to each line of $\mathcal{N}$, we can define a unitary representation of U(1) on each index of each tensor in $\mathcal{N}$. Then we say that $\mathcal{N}$ is a U(1)-invariant tensor network if all its tensors are U(1) invariant. Notice that, by construction, if $\mathcal{N}$ is a U(1)-invariant tensor network, then the resulting linear map $\hat{T}$ is also U(1) invariant. This is illustrated in Fig. 11.

More generally, we can consider a U(1)-symmetric tensor network, made of tensors that are U(1) symmetric (that is, either invariant or covariant). Recall, however, that any covariant tensor can be represented as an invariant tensor by adding an extra index (80). Therefore without loss of generality we can restrict our attention to invariant tensor networks.

D. Tensor network states and algorithms with U(1) symmetry

As discussed in Sec. II E, a tensor network $\mathcal{N}$ can be used to describe certain pure states $|\Psi\rangle \in \mathbb{V}^{\otimes L}$ of a lattice $\mathcal{L}$. If $\mathcal{N}$ is a U(1)-symmetric tensor network then it will describe a pure state $|\Psi\rangle$ that has a well-defined total particle number N. That is, a U(1)-symmetric pure state

$$\hat{N}|\Psi\rangle = N|\Psi\rangle, \quad e^{-i\hat{N}\varphi}|\Psi\rangle = e^{-iN\varphi}|\Psi\rangle. \tag{85}$$

In this way we can obtain a more refined version of popular tensor network states such as MPS, TTN, MERA, PEPS, etc. As a variational *Ansatz*, a symmetric tensor network state is more constrained than a regular tensor network state, and consequently it can represent fewer states $|\Psi\rangle \in \mathbb{V}^{\otimes L}$. However, it also depends on fewer parameters. This implies a more economical description, as well as the possibility of reducing computational costs during its manipulation.

The rest of this section is devoted to explaining how one can achieve a reduction in computational costs. This is based on storing and manipulating U(1)-invariant tensors expressed in the canonical form of Eqs. (83) and (84). We next explain how to adapt the set $\mathcal{P}$ of four primitive operations for the tensor network manipulations discussed in Sec. II D, namely, permutation and reshaping of indices, matrix multiplication, and factorization.

E. Permutation of indices

Given a U(1)-invariant tensor $\hat{T}$ expressed in the canonical form of Eqs. (83) and (84), permuting two of its indices is straightforward. It is achieved by swapping the position of the two particle numbers of $\hat{T}_{n_1 n_2 \cdots n_k}$ involved, and also the corresponding degeneracy indices. For instance, if the rank-3 tensor $\hat{T}$ of Eq. (16) is U(1) invariant and has components

$$\hat{T}_{abc} = \left(\hat{T}_{n_A n_B n_C}\right)_{t_{n_A} t_{n_B} t_{n_C}} \delta_{n_A + n_B, n_C} \tag{86}$$

when expressed in the particles number basis $a = (n_A, t_{n_A})$, $b = (n_B, t_{n_B})$, $c = (n_C, t_{n_C})$, then tensor $\hat{T}'$ of Eq. (1), obtained from $\hat{T}$ by permuting the last two indices, has components

$$(\hat{T}')_{acb} = \left(\hat{T}'_{n_A n_C n_B}\right)_{t_{n_A} t_{n_C} t_{n_B}} \delta_{n_A + n_B, n_C}, \tag{87}$$

where

$$\left(\hat{T}'_{n_A n_C n_B}\right)_{t_{n_A} t_{n_C} t_{n_B}} = \left(\hat{T}_{n_A n_B n_C}\right)_{t_{n_A} t_{n_B} t_{n_C}}. \tag{88}$$

Notice that since we only need to permute the components of those $\hat{T}_{n_A n_B n_C}$ such that $n_A + n_B = n_C$, implementing the permutation of indices requires less computation time than a regular index permutation. This is shown in Fig. 12, corresponding to a permutation of indices using MATLAB.

F. Reshaping of indices

The indices of a U(1)-invariant tensor can be reshaped (fused or split) in a similar manner to those of a regular tensor. However, maintaining the convenient canonical form of Eqs. (83) and (84) requires additional steps. Two adjacent indices can be fused together using the table Υ^{fuse} of Eq. (45), which is a sparse tensor made of 1's and 0's. Similarly an index can be split into two adjacent indices by using its inverse, the sparse tensor Υ^{split} of Eq. (48).

Example 8. Let us consider again the rank-3 tensor $\hat{T}$ of Eq. (16) with components given by Eq. (86), where a and b are outgoing indices and c is an incoming index. We can fuse outgoing index b and incoming index c into an (e.g., incoming) index d, obtaining a new tensor $\hat{T}'$ with components

$$(\hat{T}')_{ad} = \left(\hat{T}'_{n_A n_D}\right)_{t_{n_A} t_{n_D}} \delta_{n_A, n_D}, \tag{89}$$

where $n_D = -n_B + n_C$. (The sign in front of n_B comes from the fact that d is an incoming index and b is an outgoing index.) The components of $\hat{T}'$ are in one-to-one correspondence with those of $\hat{T}$ and follow from the transformation

$$\left(\hat{T}'_{n_A n_D}\right)_{t_{n_A} t_{n_D}} = \sum_{n_B, t_{n_B}, n_C, t_{n_C}} \left(\hat{T}_{n_A n_B n_C}\right)_{t_{n_A} t_{n_B} t_{n_C}}$$
$$\times \Upsilon^{\text{fuse}}_{n_B t_{n_B}, n_C t_{n_C} \rightarrow n_D t_{n_D}}, \tag{90}$$

where only the case $n_A = n_D$ needs to be considered. To complete the example, let us assume that the index a is described by the vector space $\mathbb{V}^{(A)} \cong \mathbb{V}_0 \oplus \mathbb{V}_1 \oplus \mathbb{V}_2$ with degeneracies $d_0 = 1$, $d_1 = 2$, and $d_2 = 1$; index b is described by a vector space $\mathbb{V}^{(B)} \cong \mathbb{V}_{-1} \oplus \mathbb{V}_0$ without degeneracies, i.e., $d_{-1} = d_0 = 1$; and index c is described by a vector space $\mathbb{V}^{(C)} \cong \mathbb{V}_0 \oplus \mathbb{V}_1$ also without degeneracies, $d_{-1} = d_0 = 1$.
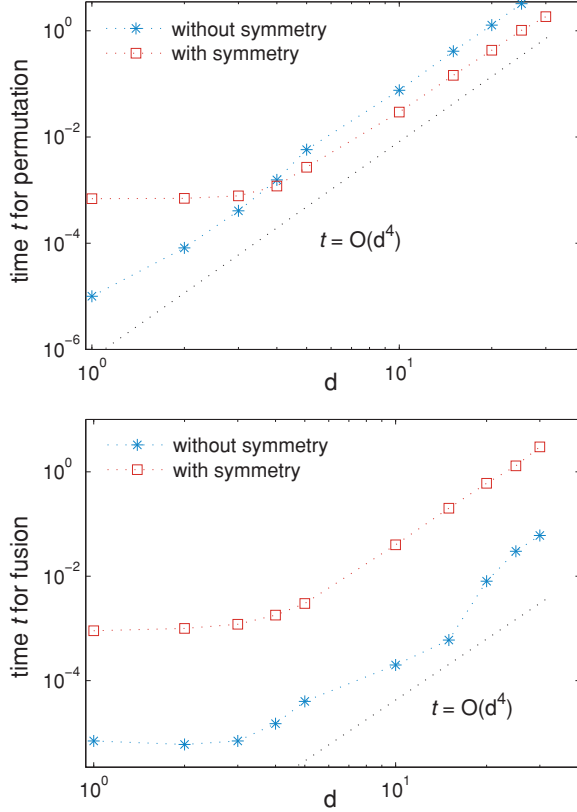
FIG. 12. (Color online) Computation times (in seconds) required to permute and fuse two indices of a rank-4 tensor $\hat{T}$, as a function of the size of the indices. All four indices of $\hat{T}$ have the same size, $5d$, and therefore the tensor contains $|\hat{T}| = 5^4 d^4$ coefficients. The figures compare the time required to perform these operations using a regular tensor and a U(1)-invariant tensor, where in the second case each index contains five different values of the particle number n (each with degeneracy d) and the canonical form of Eqs. (83) and (84) is used. The upper figure shows the time required to permute two indices: For large d, exploiting the symmetry of a U(1)-invariant tensor by using the canonical form results in shorter computation times. The lower figure shows the time required to fuse two adjacent indices. In this case, maintaining the canonical form requires more computation time. Notice that in both figures the asymptotic cost scales as $O(d^4)$, or the size of $\hat{T}$, since this is the number of coefficients which need to be rearranged. We note that the fixed-cost overheads associated with symmetric manipulations could potentially vary substantially with choice of programming language, compiler, and machine architecture. The results given here show the performance of the authors' MATLAB implementation of U(1) symmetry.

Then $\mathbb{V}^{(D)} \cong \mathbb{V}^{(B)} \otimes \mathbb{V}^{(C)}$ (and in this example, also $\mathbb{V}^{(D)} \cong \mathbb{V}^{(A)}$) and Eq. (90) amounts to

$$\left(\hat{T}'_{00}\right)_{11} = \left(\hat{T}_{000}\right)_{111}, \quad \left(\hat{T}'_{11}\right)_{11} = \left(\hat{T}_{101}\right)_{111},$$

$$\left(\hat{T}'_{11}\right)_{12} = \left(\hat{T}_{101}\right)_{211}, \quad \left(\hat{T}'_{11}\right)_{21} = \left(\hat{T}_{1(-1)0}\right)_{111},$$

$$\left(\hat{T}'_{11}\right)_{22} = \left(\hat{T}_{1(-1)0}\right)_{211}, \quad \left(\hat{T}'_{22}\right)_{11} = \left(\hat{T}_{2(-1)1}\right)_{111},$$

where we notice that tensor $\hat{T}'$ is a matrix as in Eq. (38). Similarly, we can split incoming index d of tensor $\hat{T}'$ back into

outgoing index b and incoming index c of tensor $\hat{T}$ according to

$$\left(\hat{T}_{n_A n_B n_C}\right)_{t_{n_A} t_{n_B} t_{n_C}} = \sum_{n_D, t_{n_D}} \left(\hat{T}'_{n_A n_D}\right)_{t_{n_A} t_{n_D}}$$
$$\times \Upsilon^{\text{split}}_{n_D t_{n_D} \to n_B t_{n_B}, n_C t_{n_C}}, \quad (91)$$

which, again, is nontrivial only for $-n_B + n_C = n_D$ and $n_A + n_B = n_C$.

This example illustrates that fusing and splitting indices while maintaining the canonical form of Eqs. (83) and (84) requires more work than reshaping regular indices. Indeed, after taking indices b and c into $d = b \times c$ by listing all pairs of values $b \times c$, we still need to reorganize the resulting basis elements according to their particle number n_D. Although this can be done by following the simple table given by Υ^{fuse}, it may add significantly to the overall computational cost associated with reshaping a tensor. For instance, Fig. 12 shows that fusing indices of invariant tensors can be more expensive than fusing indices of regular tensors.

G. Multiplication of two matrices

By permuting and reshaping the indices of a U(1)-invariant tensor, we can convert it into a U(1)-invariant matrix $\hat{T} = \bigoplus_{nn'} \hat{T}_{nn'} \delta_{n,n'}$, or simply

$$\hat{T} = \bigoplus_n \hat{T}_n, \quad (92)$$

where $\hat{T}_n \equiv \hat{T}_{nn}$. In components, matrix $\hat{T}$ reads

$$\left(\hat{T}\right)_{ab} = \left(\hat{T}_n\right)_{t_n t'_n}, \quad (93)$$

where $a = (n, t_n)$ and $b = (n, t'_n)$. In particular, similar to the discussion in Sec. II B for regular tensors, the multiplication of two tensors invariant under the action of U(1) can be reduced to the multiplication of two U(1)-invariant matrices.

Let $\hat{R}$ and $\hat{S}$ be two U(1)-invariant matrices, with canonical forms

$$\hat{R} = \bigoplus_n \hat{R}_n, \quad \hat{S} = \bigoplus_n \hat{S}_n. \quad (94)$$

Their product $\hat{T} = \hat{R} \cdot \hat{S}$, Eq. (4), is then another matrix $\hat{T}$ which is also block diagonal,

$$\hat{T} = \bigoplus_n \hat{T}_n, \quad (95)$$

such that each block $\hat{T}_n$ is obtained by multiplying the corresponding blocks $\hat{R}_n$ and $\hat{S}_n$,

$$\hat{T}_n = \hat{R}_n \cdot \hat{S}_n. \quad (96)$$

Equations (92) and (96) make evident the potential reduction of computational costs that can be achieved by manipulating U(1)-invariant matrices in their canonical form. First, a reduction in memory space follows from only having to store the diagonal blocks in Eq. (92). Second, a reduction in computational time is implied by only having to multiply these blocks in Eq. (96). This is illustrated in the following example.

Example 9. Consider a U(1)-invariant matrix $\hat{T}$ which is a linear map in a space $\mathbb{V}$ that decomposes into q irreps $\mathbb{V}_n$, each

of which has the same degeneracy $d_n = d$. That is, $\hat{T}$ is a square matrix of dimensions $dq \times dq$, with the block-diagonal form of Eq. (92). Since there are q blocks $\hat{T}_n$ and each block has size $d \times d$, the U(1)-invariant matrix $\hat{T}$ contains qd^2 coefficients. For comparison, a regular matrix of the same size contains q^2d^2 coefficients, a number greater by a factor of q.

Let us now consider multiplying two such matrices. We use an algorithm that requires $O(l^3)$ computational time to multiply two matrices of size $l \times l$. The cost of performing q multiplications of $d \times d$ blocks in Eq. (96) scales as $O(qd^3)$. In contrast, the cost of multiplying two regular matrices of the same size scales as $O(q^3d^3)$, requiring q^2 times more computation time. Figure 13 shows a comparison of computation times when multiplying two matrices for both U(1)-symmetric and regular matrices.

H. Factorization of a matrix

The factorization of a U(1)-invariant matrix $\hat{T}$ [Eq. (92)] can also benefit from the block-diagonal structure. Consider, for instance, the singular value decomposition $\hat{T} = \hat{U}\hat{S}\hat{V}$ of Eq. (11). In this case we can obtain the matrices

$$\hat{U} = \bigoplus_n \hat{U}_n, \quad \hat{S} = \bigoplus_n \hat{S}_n, \quad \hat{V} = \bigoplus_n \hat{V}_n \qquad (97)$$

by performing the singular value decomposition of each block $\hat{T}_n$ independently,

$$\hat{T}_n = \hat{U}_n \hat{S}_n \hat{V}_n. \qquad (98)$$

The computational savings are analogous to those described in example 9 above for the multiplication of matrices. Figure 13 also shows a comparison of computation times required to perform a singular value decomposition on U(1)-invariant and regular matrices using MATLAB.

I. Discussion

In this section we have seen that U(1)-invariant tensors can be written in the canonical form of Eqs. (83) and (84), and that this canonical form is of interest because it offers a compact description in terms of only those coefficients which are not constrained by the symmetry. We have also seen that maintaining the canonical form during tensor manipulations adds some computational overhead when reshaping (fusing or splitting) indices, but reduces computation time when permuting indices (for sufficiently large tensors) and when multiplying or factorizing matrices (for sufficiently large matrix sizes).

The cost of reshaping and permuting indices is proportional to the size $|\hat{T}|$ of the tensors, whereas the cost of multiplying and factorizing matrices is a larger power of the matrix size, for example, $|\hat{T}|^{3/2}$. The use of the canonical form when manipulating large tensors therefore frequently results in an overall reduction in computation time, making it a very attractive option in the context of tensor network algorithms. This is exemplified in the next section, where we apply the MERA to study the ground state of quantum spin models with a U(1) symmetry.

On the other hand, however, the cost of maintaining invariant tensors in the canonical form becomes more relevant
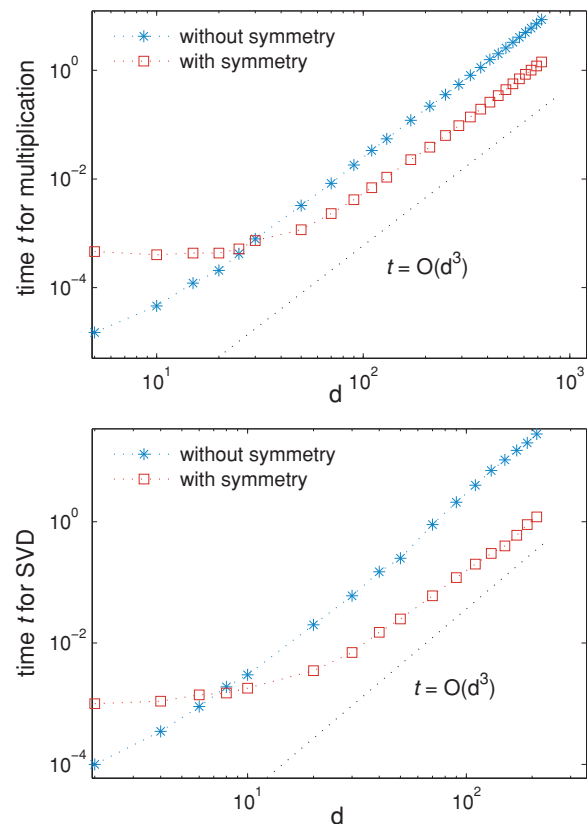

FIG. 13. (Color online) Computation times (in seconds) required to multiply two matrices (upper panel) and to perform a singular value decomposition (lower panel) as a function of the size of the indices. Matrices of size $5d \times 5d$ are considered. The figures compare the time required to perform these operations using regular matrices and U(1)-invariant matrices, where for the U(1)-invariant matrices each index contains five different values of the particle number n, each with degeneracy d, and the canonical form of Eqs. (92) and (93) is used. That is, each matrix decomposes into five blocks of size $d \times d$. For large d, exploiting the block-diagonal form of U(1)-invariant matrices results in shorter computation time both for multiplication and for singular value decomposition. The asymptotic cost scales with d as $O(d^3)$, while the size of the matrices grows as $O(d^2)$. We note that the fixed-cost overheads associated with symmetric manipulations could potentially vary substantially with choice of programming language, compiler, and machine architecture. The results given here show the performance of the authors' MATLAB implementation of U(1) symmetry.

when dealing with smaller tensors. In the next section we will also see that in some situations, this additional cost may significantly reduce, or even offset, the benefits of using the canonical form. In this event, and in the specific context of algorithms where the same tensor manipulations are iterated many times, it is possible to significantly decrease the additional cost by *precomputing* the parts of the tensor manipulations that are repeated on each iteration. Precomputation schemes are described in more detail in the Appendix. Their performance is illustrated in the next section.

V. TENSOR NETWORK ALGORITHMS WITH U(1) SYMMETRY: A PRACTICAL EXAMPLE

In previous sections we have described a strategy to incorporate a U(1) symmetry into tensors, tensor networks, and their manipulations. To further illustrate how the strategy works in practice, in this section we demonstrate its use in the context of the multiscale entanglement renormalization *Ansatz*, or MERA, and present numerical results from our reference implementation of the U(1) symmetry in MATLAB.

A. Multiscale entanglement renormalization *Ansatz*

Figure 14 shows a MERA that represent states $|\Psi\rangle \in \mathbb{V}^{\otimes L}$ of a lattice $\mathcal{L}$ made of $L = 18$ sites (see Sec. II E). Recall that the MERA is made of layers of isometric tensors, known as disentanglers $\hat{u}$ and isometries $\hat{w}$, that implement a coarse-graining transformation. In this particular scheme, isometries map three sites into one and the coarse-graining transformation reduces the $L = 18$ sites of $\mathcal{L}$ into two sites using two layers of tensors. A collection of states on these two sites is then encoded in a top tensor $\hat{t}$, whose upper index $a = 1, 2, \ldots, \chi_{\text{top}}$ is used to label χ_{top} states $|\Psi_a\rangle \in \mathbb{V}^{\otimes L}$. This particular arrangement of tensors corresponds to the 3:1 MERA described in Ref. 18.

In this section we will consider a MERA analogous to that of Fig. 14 but with Q layers of disentanglers and isometries, which we will use to describe states on a lattice $\mathcal{L}$ made of 2×3^Q sites. We will use this variational *Ansatz* to obtain an approximation to the ground state and first excited states of two quantum spin chains that have a global internal U(1) symmetry, namely the spin-1/2 quantum XX chain of Eq. (67) and the spin-1/2 antiferromagnetic quantum Heisenberg chain of Eq. (68). Each spin-1/2 degree of freedom of the chain is described by a vector space spanned by two orthonormal states $\{|\downarrow\rangle, |\uparrow\rangle\}$. Here we will represent them by the states $\{|0\rangle, |1\rangle\}$ corresponding to zero and one particles, as in example 1 of Sec. III A. For computational convenience, we will consider a lattice $\mathcal{L}$ where each site contains two spins, or states, $\{|\downarrow\downarrow\rangle, |\downarrow\uparrow\rangle, |\uparrow\downarrow\rangle, |\uparrow\uparrow\rangle\}$. Therefore each site of $\mathcal{L}$ is described by a space $\mathbb{V} \cong \mathbb{V}_0 \oplus \mathbb{V}_1 \oplus \mathbb{V}_2$, where $d_0 = d_2 = 1$ and $d_1 = 2$, as in example 2 of Sec. III A. Thus a lattice $\mathcal{L}$ made of L sites corresponds to a chain of $2L$ spins. In such a system, the total particle number N ranges from 0 to $2L$. (Equivalently, the z component of the total spin S_z ranges from $-L$ to L, with $S_z = N - L$.)
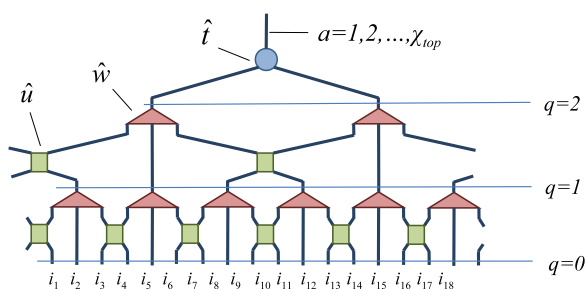

FIG. 14. (Color online) MERA for a system of $L = 2 \times 3^2 = 18$ sites, made of two layers of disentanglers $\hat{u}$ and isometries $\hat{w}$, and a top tensor $\hat{t}$.

B. MERA with U(1) symmetry

A U(1)-invariant version of the MERA, or U(1) MERA for short, is obtained by simply considering U(1)-invariant versions of all of the isometric tensors, namely the disentanglers $\hat{u}$, isometries $\hat{w}$, and the top tensor $\hat{t}$. This requires assigning a particle number operator to each index of the MERA. Each open index of the first layer of disentanglers corresponds to one site of $\mathcal{L}$. The particle number operator on any such index is therefore given by the quantum spin model under consideration. We can characterize the particle number operator by two vectors, $\vec{n}$ and $\vec{d}$: a list of the different values the particle number takes and the degeneracy associated with each such particle number, respectively. In the case of the vector space $\mathbb{V}$ for each site of $\mathcal{L}$ described above, $\vec{n} = \{0, 1, 2\}$ and $\vec{d} = \{1, 2, 1\}$. For the open index of the tensor $\hat{t}$ at the very top of the MERA, the assignment of charges is also straightforward. For instance, to find an approximation to the ground state and first seven excited states of the quantum spin model with particle number N, we choose $\vec{n} = \{N\}$ and $\vec{d} = \{8\}$. (In particular, a vanishing S_z corresponds to $N = L$.)

For each of the remaining indices of the MERA, the assignment of the pair $(\vec{n}, \vec{d})$ needs careful consideration and a final choice may only be possible after numerically testing several options and selecting the one which produces the lowest expectation value of the energy. Table I shows the assignment of particle numbers and degeneracies made to represent the ground state and several excited states in a system of $L = 2 \times 3^3 = 54$ sites (that is, 108 spins) with total particle number $N = L = 54$ (or $S_z = 0$). Notice that at level q of the MERA ($q = 1, 2, 3$), each index effectively corresponds to a block of $n_q \equiv 3^q$ sites of $\mathcal{L}$. Therefore having exactly n_q particles in a block of n_q sites corresponds to a density of one particle per site of $\mathcal{L}$. The assigned particle numbers of Table I, namely, $[n_q - 2, n_q - 1, n_q, n_q + 1, n_q + 2]$ for level q, then correspond to allowing for fluctuations of up to two particles with respect to the average density. The sum of corresponding degeneracies $\vec{d} = \{d_{n_q-2}, d_{n_q-1}, d_{n_q}, d_{n_q+1}, d_{n_q+2}\}$ gives the bond dimension χ, which in the example is $\chi = 13$.

In order to find an approximation to the ground state of either $\hat{H}_{XX}$ or $\hat{H}_{XXX}$ in Eqs. (67) and (68), we set $\chi_{\text{top}} = 1$ and optimize the tensors in the MERA so as to minimize the expectation value

$$\langle \Psi | \hat{H} | \Psi \rangle, \qquad (99)$$

where $|\Psi\rangle \in \mathbb{V}^{\otimes L}$ is the pure state represented by the MERA and $\hat{H}$ is the relevant Hamiltonian. In order to find an

TABLE I. Example of particle number assignment in a U(1) MERA for $L = 54$ sites (or 108 spins). The total bond dimension is $\chi = 1 + 3 + 5 + 3 + 1 = 13$. The value of χ_{top} is set as described in the text.

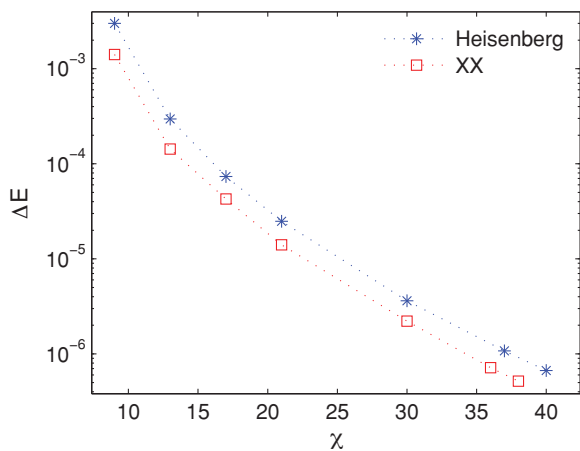| Level q	Particle numbers $\vec{n}$	Degeneracies $\vec{d}$
top	$\{N = 54\}$	$\{\chi_{\text{top}}\}$
3	$\{25, 26, 27, 28, 29\}$	$\{1, 3, 5, 3, 1\}$
2	$\{7, 8, 9, 10, 11\}$	$\{1, 3, 5, 3, 1\}$
1	$\{1, 2, 3, 4, 5\}$	$\{1, 3, 5, 3, 1\}$
0	$\{0, 1, 2\}$	$\{1, 2, 1\}$

FIG. 15. (Color online) Error in ground-state energy ΔE as a function of χ for the XX and Heisenberg models with $2L = 108$ spins and periodic boundary conditions, in the particle number sector $N = L$ (or $S_z = 0$). The error is calculated with respect to the exact solutions, and is seen to decay exponentially with χ.

approximation to the $\chi_{\mathrm{top}} > 1$ eigenstates of $\hat{H}$ with lowest energies, we optimize the tensors in the MERA so as to minimize the expectation value

$$\sum_{a=1}^{\chi_{\mathrm{top}}} \langle \Psi_a | \hat{H} | \Psi_a \rangle, \quad \langle \Psi_a | \Psi_{a'} \rangle = \delta_{aa'}. \qquad (100)$$

The optimization is carried out using the MERA algorithm described in Ref. [18], which requires contracting tensor networks (by sequentially multiplying pairs of tensors) and performing singular value decompositions. In the present example, all of these operations will be performed exploiting the U(1) symmetry.

Figure 15 shows the error in the ground-state energy as a function of the bond dimension χ, for assignments of degeneracies similar to those in Table II. The error is seen to decay exponentially with increasing χ, indicating increasingly accurate approximations to the ground state.

TABLE II. Number of coefficients required to specify the tensors of a MERA for $L = 54$ as a function of the bond dimension χ, decomposed according to a degeneracy vector $\vec{d}$. A comparison is made between regular tensors and U(1)-invariant tensors.

χ	Degeneracy $\vec{d}$	No. of coefficients (regular)	No. of coefficients (symmetric)	Ratio
4	[0,1,2,1,0]	1552	426	3.6 : 1
8	[0,2,4,2,0]	17216	4714	3.7 : 1
13	[1,3,5,3,1]	115501	21969	5.3 : 1
17	[1,4,7,4,1]	335717	68469	5.0 : 1
21	[1,5,9,5,1]	779965	166901	4.7 : 1
30	[2,7,12,7,2]	3243076	639794	5.1 : 1

C. Exploiting the symmetry

We now discuss some of the advantages of using the U(1) MERA.

1. Selection of particle number sector

An important advantage of the U(1) MERA is that it exactly preserves the U(1) symmetry. In other words, the states resulting from a numerical optimization are exact eigenvectors of the total particle number operator $\hat{N}$ [Eq. (57)]. In addition, the total particle number N can be preselected at the onset of optimization by specifying it in the open index of the top tensor $\hat{t}$.

Figure 16 shows the energy gap between the ground state and two excited states of an XX chain with $2L$ spins (or L sites), for $N = L$ particles ($S_z = 0$). One is the first excited state which also has $N = L$ particles. The other is the ground state in the sector with $N = L + 1$ particles. The two energy gaps are seen to decay with the system size as L^{-1}. The ability to preselect a given particle number N means that only two optimizations were required: one MERA optimization for $N = L$ with $\chi_{\mathrm{top}} = 2$ in order to obtain an approximation to the ground state and first excited state of $\hat{H}_{XX}$ in that particle number sector; and one MERA optimization for $N = L + 1$ with $\chi_{\mathrm{top}} = 1$ in order to obtain an approximation to the ground state of $\hat{H}_{XX}$ in the particle number sector $N = L + 1$.

Similar results can be obtained with the regular MERA. For instance, one can obtain an approximation to the ground state of a given particle number sector by adding a chemical potential term $-\mu \sum_s \hat{n}^{(s)}$ to the Hamiltonian and carefully tuning the chemical potential term μ until the expectation value of the particle number $\hat{N}$ is the desired one. However, the regular MERA cannot guarantee that the states obtained in this way are exact eigenvectors of $\hat{N}$. Instead the resulting states are likely to have particle number fluctuations.

Figure 17 shows the low-energy spectrum of the Heisenberg model $\hat{H}_{XXX}$ for a periodic system of $L = 54$ sites (or
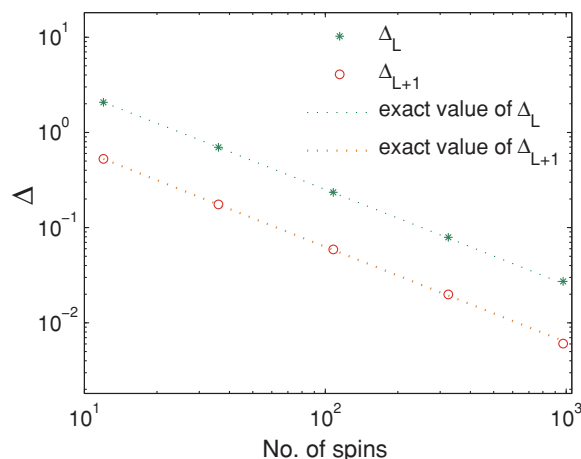

FIG. 16. (Color online) Decay of energy gaps Δ with system size L in the XX model. The upper line corresponds to the energy gap Δ_L between the ground state and the first excited state in the $N = L$ particle number (or $S_z = 0$) sector. The lower line corresponds to the energy gap Δ_{L+1} between the ground states of the $N = L$ and $N = L + 1$ particle number sectors.
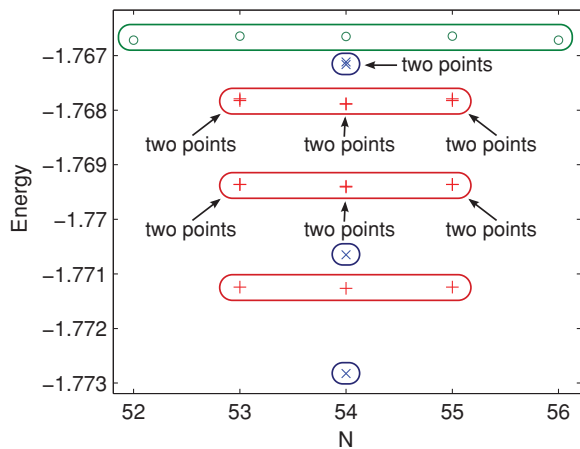
FIG. 17. (Color online) Low-energy spectrum of $\hat{H}_{XXX}$ with $L = 54$ sites ($= 108$ spins). Depicted states have spins of zero ($\times$, blue loops), one ($+$, red loops), or two ($\circ$, green loop), and total number of particles (N) between 52 and 56. Note that the second and third spin-1 triplets are twofold degenerate.
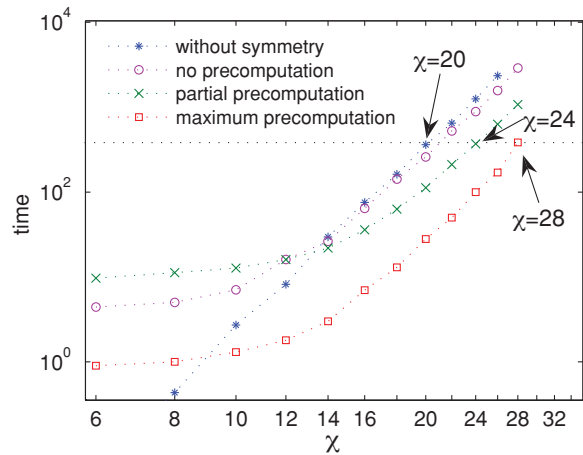

FIG. 18. (Color online) Computation time (in seconds) for one iteration of the MERA energy minimization algorithm, as a function of the bond dimension χ. For sufficiently large χ, exploiting the U(1) symmetry leads to reductions in computation time. The horizontal line on this graph shows that this reduction in computation time equates to the ability to evaluate MERAs with a higher bond dimension χ: For the same cost per iteration incurred when optimizing a standard MERA in MATLAB with bond dimension $\chi = 20$, one may choose instead to optimize a U(1)-symmetric MERA with partial precomputation and $\chi = 24$, or with full precomputation and $\chi = 28$.

108 spins), including the ground state and several excited states both in the particle sector $N = 54$ (or $S_z = 0$), and in neighboring particle sectors. Recall that $\hat{H}_{XXX}$ is actually invariant under a global internal SU(2) symmetry, of which particle number is a U(1) subgroup. Correspondingly the spectrum is organized according to irreps of SU(2), namely singlets (total spin 0), triplets (total spin 1), quintuplets (total spin 2), etc. Again, using the U(1) MERA, the five particle number sectors $N = 52, 53, 54, 55,$ and 56 can be addressed with independent computations. This implies, for instance, that in order to find the gap between the first and fourth singlets, we can simply set $N = 54$ and $\chi_{\text{top}} = 9$ on the open index of the top tensor $\hat{t}$, to accommodate the first four spin-0 states and five spin-1 states in the $N = 54$ sector, as seen in Fig. 17. In order to capture the fourth singlet using the regular MERA, we would need to consider at least $\chi_{\text{top}} = 19$ (at a larger computational cost and possibly lower accuracy), since this state has only the 19th lowest energy overall.

2. Reduction of computational costs

The use of U(1)-invariant tensors in the MERA also results in a reduction of computational costs. First, U(1)-invariant tensors, when written in the canonical form of Eqs. (83) and (84), are block diagonal and therefore require less storage space. Table II compares the number of MERA coefficients that need to be stored in the regular and symmetric case, for different choices of particle number assignments relevant to the present examples.

Second, the computation time required to manipulate tensors is also reduced when using U(1)-invariant tensors in the canonical form. Figure 18 shows the computation time required for one iteration of the energy minimization algorithm of Ref. 18 (during which all tensors in the MERA are updated once), as a function of the total bond dimension χ. The plot compares the time required using regular tensors and U(1)-invariant tensors. For U(1)-invariant tensors, we display the time per iteration for three different levels of

precomputation, as described in the Appendix. The figure shows that for sufficiently large χ, using U(1)-invariant tensors leads to a shorter time per iteration of the optimization algorithm.

In the authors' reference implementation (written in MAT-LAB), using the symmetry without precomputation is seen to only reduce the computation time by about a factor of 2 for the largest χ under consideration. This is because maintaining the canonical form for U(1)-invariant tensors still imposes a significant overhead for the values of χ considered. In contrast, when using precomputation we obtained times shorter by a factor of 10 or more.

The magnitude of the overhead imposed by maintaining the canonical form will depend on factors such as programming language and machine architecture, but in general more significant gains can be obtained by making full use of precomputation. This option, however, requires a significant amount of additional memory (see the Appendix), and a more convenient middle ground can be obtained by using a partial precomputation scheme.

VI. CONCLUSIONS

In this paper we have provided a detailed explanation of how a global internal Abelian symmetry may be incorporated into any tensor network algorithm. Following Ref. 52 we considered tensor networks constructed from tensors which were invariant under the action of the internal symmetry, and showed how each tensor may be decomposed according to a canonical form into *degeneracy* tensors (which contain all the degrees of freedom that are not affected by the symmetry) and *structural* tensors (which are completely determined by the symmetry). We then introduced a set of primitive operations

TABLE III. Fusion rules for the group $\mathbb{Z}_3$: Outcomes of evaluating $a \times a'$.

		a		
		0	1	2
	0	0	1	2
a'	1	1	2	0
	2	2	0	1

$\mathcal{P}$ which may be used to carry out tensor network algorithms using *Ansätze* such as MPS, PEPS, and MERA, and showed how each of these operations can be implemented in such a way that the canonical form is both preserved and exploited for computational gain.

We then demonstrated the implementation of this decomposition for tensors with an internal U(1) symmetry, and computed multiple benchmarks demonstrating the computational costs and speedups inherent in this approach. We found that although maintaining the canonical form imposed additional costs when combining or splitting tensor indices, for simulations of a sufficiently large scale these costs can be offset by the gains made when performing permutations, matrix multiplications, and matrix decompositions.

Finally, we implemented the MERA on a quantum spin chain with U(1) symmetry. We showed that exploitation of this symmetry can lead to a decrease in the computational cost by a factor of between 10 and 20. These gains may be used either to reduce overall computation time or to permit substantial increases in the MERA bond dimension χ, and consequently in the accuracy of the results obtained.

Although in this paper we have focused on an example which is a continuous Abelian group, the formalism presented here may equally well be applied to a finite Abelian group. In particular let us consider a cyclic group Z_q, $q \in \mathbb{Z}^+$.[75] As in the case of U(1), the Hilbert space decomposes under the action of the group into a direct sum of one-dimensional irreps which are each characterized by an integer charge a, and consequently most of the analysis presented in this paper remains unchanged. In particular, matrices which are invariant under the action of the group will be block diagonal in the basis labeled by charge according to Eq. (35), and symmetric tensors enjoy the canonical decomposition stated in Eqs. (83) and (84). The only objects which need modification are the fusion and splitting maps, which need to be altered so that they encode the fusion rules for Z_q instead for U(1). For a cyclic group Z_q, the fusion of two charges a and a' gives rise to a charge a'' according to $a'' = (a + a')|_q$ where $|_q$ indicates that the addition is performed modulo q. For example, Z_3 has charges $a = 0,1,2$, and the fusion rules for Z_3 take the form $a \times a' \to a''$ where the value of a'' is given in Table III.

More generally, a generic Abelian group will be characterized by a set of charges $(a_1, a_2, a_3, \ldots)$. When fusing two such sets of charges $(a_1, a_2, a_3, \ldots)$ and $(a_1', a_2', a_3', \ldots)$, each charge a_i is combined with its counterpart a_i' according to the fusion rule of the relevant subgroup. Once again, this behavior may be encoded in a single fusion map Υ^{fuse} and its inverse Υ^{split}.

The formalism presented in this paper is therefore directly applicable to any Abelian group.

APPENDIX: USE OF PRECOMPUTATION IN ITERATIVE ALGORITHMS

We have seen that the use of the canonical form given in Eqs. (83) and (84) to represent U(1)-invariant tensors can potentially lead to substantial reductions in memory requirements and in calculation time. We also pointed out, however, that there is an additional cost in maintaining an invariant tensor in its canonical form, and that this is associated with the reshaping (fusing and/or splitting) of its indices. In some situations this additional cost may significantly reduce, or even offset, the benefits of using the canonical form.

In this Appendix we investigate techniques for reducing this additional cost in the context of iterative tensor network algorithms. Many of the algorithms discussed in Sec. II E are iterative algorithms, repeating the same sequence of tensor network manipulations many times over. Examples include algorithms which compute tensor network approximations to the ground state by minimizing the expectation value of the energy or by simulating evolution in imaginary time, with each iteration yielding an increasingly accurate approximation to the ground state of the system.

The goal of this Appendix is to identify calculations which depend only on the symmetry group, and are independent of the variational coefficients of such algorithms. Where these calculations are repeated in each iteration of the algorithm, we can effectively eliminate the associated computational cost by performing them only once, either during or prior to the first iteration of the algorithm, and then storing and reusing these *precomputed* results in subsequent iterations. We will illustrate this procedure by considering the precomputation of a series of operations applied to a single tensor $\hat{T}$.

To do this, we begin by revisiting the fusion and splitting tables of Sec. III C and introducing a graphical representation of these objects. We then introduce a convenient decomposition of a symmetric tensor into a matrix accompanied by multiple fusion and/or splitting tensors, and linear maps Γ that map one such decomposition into another. These linear maps are independent of the coefficients of the tensor being reorganized, and consequently they are precisely the objects which can be precomputed in order to quicken an iterative algorithm at the expense of additional memory cost. Finally we describe two specific precomputation schemes, differing in what is precomputed and in how the precomputed data are utilized during the execution of the algorithm, in order to illustrate the tradeoff between the amount of memory needed to store the precomputation data and the associated computational speedup which may be obtained. In practice, the nature of the specific implementation employed will depend on available computational resources.

1. Diagrammatic notation of fusing and splitting tensors

In describing how we can precompute repeated manipulations of this tensor $\hat{T}$, we will find it useful to employ diagrammatic representations of the fusion and splitting tables Υ^{fuse} and Υ^{split} introduced in Sec. III C. These tables implement a linear map between a pair of indices and their fusion product, and thus can be understood as trivalent tensors having two input legs and one output leg (or vice versa) in accordance with Sec. II F. We choose to represent them graphically as shown in Fig. 19(a), where the arrow within the circle always points toward the coupled index. The linear maps Υ^{fuse} and Υ^{split} are unitary, and consequently we impose that the tensors of Fig. 19(a) must satisfy the identities given in Fig. 19(b), corresponding to unitarity under the action of the conjugation operation employed in diagrammatic tensor network notation (vertical reflection of a tensor and the complex conjugation of its components, typically denoted †). Our notation also reflects the property, first noted in Sec. III C, that Υ^{fuse} and Υ^{split} may be decomposed into two pieces [Fig. 19(c)]. For the fusion tensor, we identify the first piece (represented by a circle containing an arrow) with the creation of a composed index using the manner we would employ in the absence of symmetry (2). The second piece, represented by the small square, permutes the basis elements of the composed index, reorganizing them according to total particle number. The two components of the splitting tensor are then uniquely defined by consistency with the process of conjugation for the diagrammatic representation of tensors, and with the unitarity condition of Fig. 19(b).
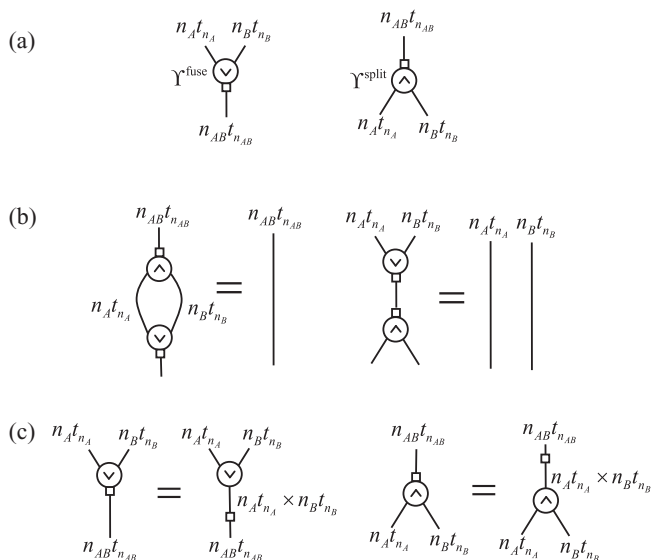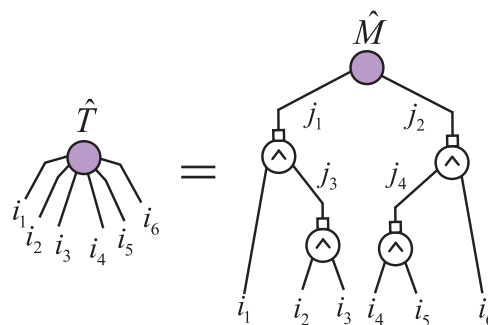

FIG. 20. (Color online) Binary tree decomposition of a symmetric tensor $\hat{T}$ having components $\hat{T}_{i_1 i_2 i_3 i_4 i_5 i_6}$. The tree $\mathcal{T}$ is comprised of a matrix $\hat{M}$ as the root node, four splitting tensors as internal nodes, and $i_1, i_2, ..., i_6$ as its leaf indices. No incoming or outgoing arrows are indicated on the indices in the figure, as the decomposition is valid for any such assignment of directional arrows.

These requirements have an important consequence. Suppose the first part of Υ^{fuse} implements $b \times c \to d$ by iterating rapidly over the values of b and more slowly over the values of c, and b lies clockwise of c on the graphical representation of Υ^{fuse}. This then means that on the graphical representation of Υ^{split} which implements $d \to b \times c$, index b must lie *counterclockwise* of c. It is therefore vitally important to distinguish between the splitting tensor and a rotated depiction of the fusing tensor. To this end we require that when using this diagrammatic notation, all tensors (with the exception of the fusion and splitting tensors) must be drawn with only downward-going legs, as seen, for example, in Fig. 20, though the legs are still free to carry either incoming or outgoing arrows as before.

2. Tree decomposition

We find it convenient to decompose a rank-k, U(1)-invariant tensor $\hat{T}$, having components $\hat{T}_{i_1 i_2 \cdots i_k}$, as a binary tree tensor network $\mathcal{T}$ consisting of a matrix $\hat{M}$ which we will call the *root node*, and of $k - 2$ splitting tensors Υ^{split} as branching *internal nodes*, with the *leaf* indices of tree $\mathcal{T}$ corresponding to the indices $\{i_1, i_2, \ldots, i_k\}$ of tensor $\hat{T}$. We refer to decomposition $\mathcal{T}$ as a tree decomposition of $\hat{T}$. Figure 20 shows an example of tree decomposition for a rank-6 tensor. It is of the form

$$\hat{T}_{i_1 i_2 i_3 i_4 i_5 i_6} = \sum_{j_1, j_2, j_3, j_4} \hat{M}_{j_1 j_2} \Upsilon^{\text{split}}_{j_1 \to i_1, j_3} \Upsilon^{\text{split}}_{j_2 \to j_4, i_6}$$

$$\times \Upsilon^{\text{split}}_{j_3 \to i_2, i_3} \Upsilon^{\text{split}}_{j_4 \to i_4, i_5}, \qquad (A1)$$

where $\{j_1, j_2, j_3, j_4\}$ are the internal indices of the tree.

The same tensor $\hat{T}$ may be decomposed as a tree in many different ways, corresponding to different choices of the fusion tree. As an example we show two different but equivalent decompositions of a rank-4 tensor in Fig. 21. Different choices $\mathcal{T}_1, \mathcal{T}_2, \ldots$ of tree decomposition for tensor $\hat{T}$ will lead to different matrix representations $\hat{M}_1, \hat{M}_2, \ldots$ of the same tensor. Finally, Fig. 22 shows how to obtain the tree decompositions from $\hat{T}_{i_1 i_2 i_3 i_4}$ by introducing an appropriate resolution of the identity, constructed from pairs of fusion operators Υ^{fuse} and splitting operators Υ^{split} in accordance with Fig. 19(b).

FIG. 19. (a) Graphical representation of the fusion tensor Υ^{fuse} and the splitting tensor Υ^{split}. (b) The tensors Υ^{fuse} and Υ^{split} are unitary, and thus yield the identity when contracted pairwise as shown. (c) A fusion tensor decomposed into two parts. The first part (indicated by a circle with an arrow) performs the tensor product of input irreps, $n_A t_A \times n_B t_B$. The result is an index that labels pairs $(n_A t_A, n_B t_B)$. The second part (indicated by a rectangle) is a permutation that associates each pair $(n_A t_A, n_B t_B)$ with a unique $(n_{AB} t_{n_{AB}})$, corresponding to a vector in the coupled basis of $\mathbb{V}^{(AB)}$.
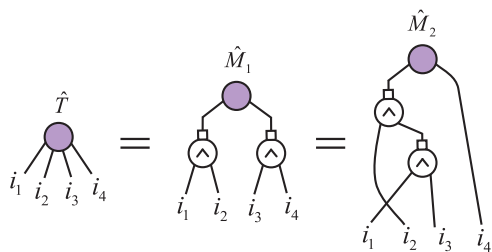
FIG. 21. (Color online) Two possible tree decompositions of a rank-4 tensor $\hat{T}$. Different choices $\mathcal{T}_1, \mathcal{T}_2, \ldots$ of tree decomposition for tensor $\hat{T}$ lead to different matrices $\hat{M}_1, \hat{M}_2, \ldots$ for the same tensor.

The representation of a tensor $\hat{T}$ by means of a tree decomposition is particularly useful because many tensor network algorithms may be understood as a sequence of operations carried out on tensors reduced to matrix form. For example, consider tensor network algorithms such as MPS, MERA, and PEPS. When tensors are updated in these algorithms, the new tensor is typically created as a matrix, to which operations from the primitive set $\mathcal{P}$ of Sec. II D are then applied. When they are decomposed or contracted with other tensors, this may once again take place with the tensor in matrix form. Any such matrix form may always be understood as the matrix component of an appropriate tree decomposition $\mathcal{T}$ of tensor $\hat{T}$, where the sequence of operations reshaping tensor $\hat{T}$ to matrix $\hat{M}$ corresponds to the contents of the shaded area in Fig. 22.
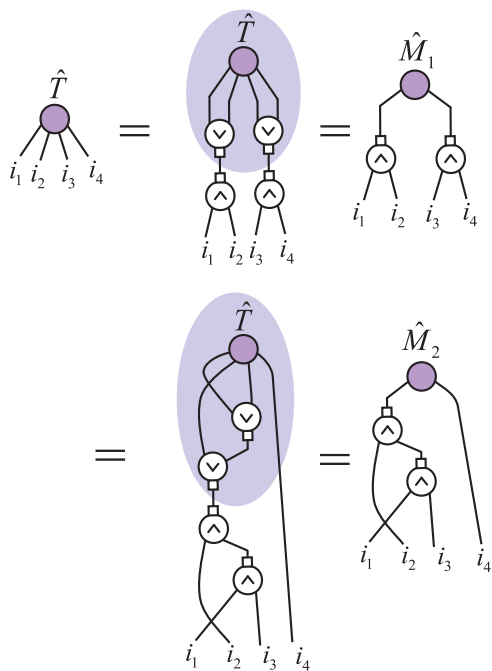

FIG. 22. (Color online) Tree decompositions of tensor $\hat{T}$ are obtained by contracting the tensor with an appropriate resolution of the identity on its indices, selected according to the desired choice of the fusion tree $\mathcal{T}$. In each instance, evaluation of the contents of the shaded region yields the appropriate matrix $\hat{M}$.
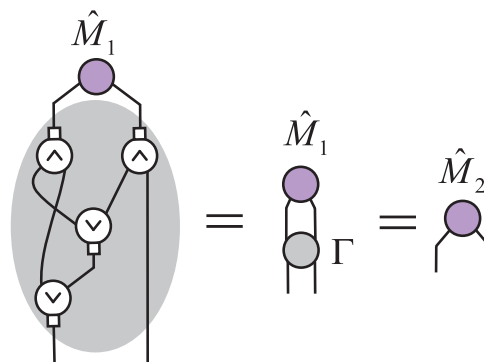

FIG. 23. (Color online) A matrix $\hat{M}_1$ can be reorganized into another matrix $\hat{M}_2$ by means of fusion tensors, splitting tensors, and the permutation of indices. These operations define a one-to-one linear map Γ that acts to reorganize the coefficients of $\hat{M}_1$. Γ does not depend on the coefficients of $\hat{M}_1$, but solely on the sequence of operations performed.

3. Mapping between tree decompositions

Suppose now that we have a tensor $\hat{T}$ in matrix form $\hat{M}_1$, which is associated with a particular choice of tree decomposition $\mathcal{T}_1$, and we wish to transform it into another matrix form $\hat{M}_2$, corresponding to another tree decomposition $\mathcal{T}_2$. As indicated, this process may frequently arise during the application of many common tensor network algorithms. The new matrix $\hat{M}_2$ can be obtained from $\hat{M}_1$ by means of a series of reshaping (splitting/fusing) and permuting operations, as indicated in Fig. 23, and this series of operations may be understood as defining a map Γ:

$$\hat{M}_2 = \Gamma(\hat{M}_1). \tag{A2}$$

The map Γ is a linear map which depends only on the tree structure of $\mathcal{T}_1$ and $\mathcal{T}_2$, and is independent of the coefficients of $\hat{M}_1$. Moreover, Γ is unitary, and it follows from the construction of fusing and splitting tensors and the behavior of permutation of indices (which serves to relocate the coefficients of a tensor) that Γ simply reorganizes the coefficients of $\hat{M}_1$ into the coefficients of $\hat{M}_2$ in a one-to-one fashion.

Therefore one way to compute the matrix $\hat{M}_2$ from matrix $\hat{M}_1$ is by first computing the linear map Γ, which is independent of the specific coefficients in tensor $\hat{T}$, and by then applying it to $\hat{M}_1$.

4. Precomputation schemes for iterative tensor network algorithms

The observation that the map Γ is independent of the specific coefficients in $\hat{M}_1$ is particularly useful in the context of iterative tensor network algorithms. It implies that, although the coefficients in $\hat{M}_1$ will change from iteration to iteration, the linear map Γ in Eq. (A2) remains unchanged. It is therefore possible to calculate the map Γ once, during the first iteration of the simulation, and then to store it in memory and reuse it during subsequent iterations. We refer to such a strategy as a *precomputation scheme*. Figure 24 contrasts the program flow of a generic iterative tensor network algorithm with and without precomputation of the transformations Γ.
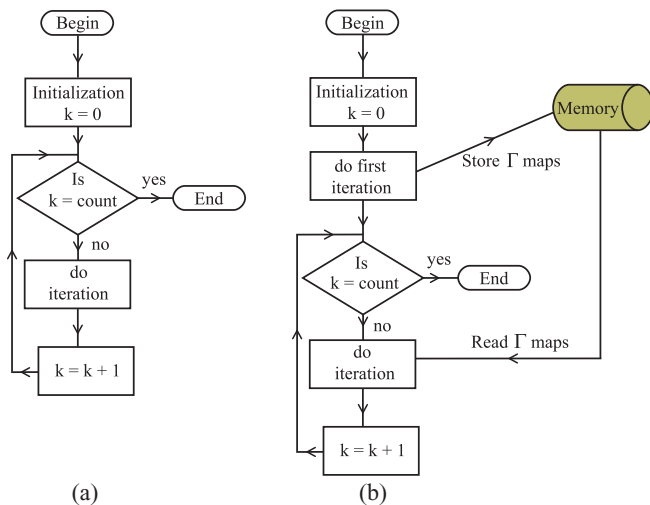
FIG. 24. (Color online) Flow diagram for the execution of a predetermined number of iterations of a generic iterative tensor network algorithm, (a) without any precomputation and (b) with precomputation of the operations Γ.

Using such a precomputation scheme, a significant speedup of simulations can be obtained, at the price of storing potentially large amounts of precomputed data (as a single iteration of the algorithm may require the application of many different transformations Γ). Therefore a tradeoff necessarily exists between the amount of speedup that can be obtained and the memory requirement that this entails. In this section we describe two different precomputation schemes. The first one fully precomputes and stores all maps Γ, and is relatively straightforward to implement. This results in the maximal increase in simulation speed, but implementation requires a large amount of memory. The second scheme only partially precomputes the maps Γ, resulting in a moderate speedup of simulations, but with memory requirements which are also similarly more modest.

a. Maximal precomputation scheme

As noted in Sec. III of this Appendix, applying the map Γ to a matrix $\hat{M}_1$ simply reorganizes its coefficients to produce the matrix $\hat{M}_2$. Moreover, if the indices of matrices $\hat{M}_1$ and $\hat{M}_2$ are fused to yield vectors $\hat{V}_1$ and $\hat{V}_2$ then the map Γ may be understood as a permutation matrix, and this in turn may be concisely represented as a string of integers $\Gamma = \gamma_1, \dots, \gamma_{|\hat{M}_1|}$ such that entry i of $\hat{V}_2 = \Gamma \hat{V}_1$ is given by entry γ_i of vector $\hat{V}_1$. Because all of the elements from which Γ is composed are sparse, unitary, and composed entirely of 0's and 1's, the permutation to which Γ corresponds may be calculated at a total cost of only $O(|\hat{M}_1|)$, where $|\hat{M}_1|$ counts only the elements of $\hat{M}_1$ which are not fixed to be zero by the symmetry constraints of Eq. (83). In essence, for each element of the vector $\hat{V}_1$ one identifies the corresponding number and degeneracy indices $(n_i^{\hat{M}_1}, t_i^{\hat{M}_1})$ on each leg $i \in \{1,2\}$ of matrix $\hat{M}_1$. One can then read down the figure, applying each table Υ^{fuse} or Υ^{split} in turn to identify the corresponding labels (n', t') on the intermediate legs, until finally the corresponding labels on the indices of $\hat{M}_2$ are obtained. There is then a further 1:1 mapping from each set of labels $(n_1^{\hat{M}_2}, t_1^{\hat{M}_2})$, $(n_2^{\hat{M}_2}, t_2^{\hat{M}_2})$ on $\hat{M}_2$

to the corresponding entry in $\hat{V}_2$, completing the definition of Γ as a map from $\hat{V}_1$ to $\hat{V}_2$.

Storing the map Γ for a transformation such as the one shown in Fig. 23 imposes a memory cost of $O(|\hat{M}_1|)$. The application of this map also incurs a computational cost of $O(|\hat{M}_1|)$, but computational overhead is saved in not having to reconstruct the map Γ on every iteration of the algorithm.

b. Partial precomputation scheme

The $O(|\hat{M}_1|)$ memory cost incurred in the previous scheme can be significant for large matrices. However, we may reduce this cost by replacing the single permutation Γ employed in that scheme with multiple smaller operations which may also be precomputed. In this approach $\hat{M}_1$ is retained in matrix form rather than being reshaped into a vector, and we precompute permutations to be performed on its rows and columns.

First, we decompose all the fusion and splitting tensors into two pieces in accordance with Fig. 19(c). Next, we recognize that any permutations applied to one or more legs of a fusion or splitting tensor may always be written as a single permutation applied to the coupled index [Fig. 25(a)]. We use this to replace all permutations on the intermediate indices of the diagram with equivalent permutations acting only on the indices of $\hat{M}_1$ and the open indices, as shown for a simple example in Fig. 25(b). The residual fusion and splitting operations, depicted by just a circle enclosing an arrow, then simply carry out fusion and splitting of indices as would be performed in the absence of symmetry (2), (3). These operations are typically far faster than their symmetric counterparts as they do not need to sort the entries of their output indices according to particle number.
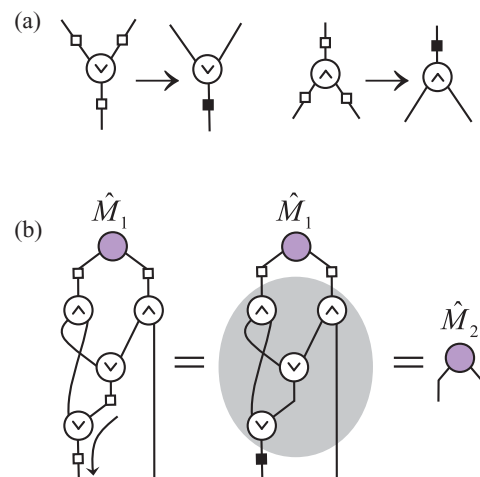

FIG. 25. (Color online) (a) Permutations applied to one or more legs of a fusion or splitting tensor can be replaced by an appropriate permutation on the coupled index. This process can be used to replace all permutations applied on internal indices of a diagram such as Fig. 23 with net permutations on the indices of $\hat{M}_1$ and on the open indices of the network, as in shown in (b). The residual fusion and splitting operations, depicted as an arrow in a circle, simply perform the basic tensor product operation and its inverse, Eqs. (2) and (3), as described in Fig. 19(c) and Sec. I of this Appendix.

In subsequent iterations, the matrix $\hat{M}_2$ is obtained from $\hat{M}_1$ by consecutively

(1) permuting the rows and columns of $\hat{M}_1$ using the precomputed net permutations which act on the legs of $\hat{M}_1$;

(2) performing any elementary (nonsymmetric) splitting, permuting of indices, and fusing operations, as described by the gray-shaded region in Fig. 25(b);

(3) permuting the rows and columns of the resulting matrix, using the precomputed net permutations which act on the open legs of Fig. 25(b).

When matrix $\hat{M}_1$ is defined compactly, as in Eq. (83), so that elements which are identically zero by symmetry are not explicitly stored, a tensor $\hat{T}$ is constructed from multiple blocks identified by U(1) charge labels on their indices [$\hat{T}_{n_1 n_2 ... n_k}$ in Eq. (83)]. Under these conditions the elementary splitting, fusing, and permutation operations of step (2) above are applied to each individual block, but some additional computational overhead is incurred in determining the necessary rearrangements of these blocks arising out of the actions performed. This rearrangement may be computed on the fly, or may also be precomputed as a mapping between the arrangement of blocks in $\hat{M}_1$ and that in $\hat{M}_2$.

The memory required to store the precomputation data in this scheme is dominated by the size of the net permutations collected on the matrix indices, and is therefore of $O(\sqrt{|\hat{M}_1|})$. The overall cost of obtaining $\hat{M}_2$ from $\hat{M}_1$ is once again of $O(|\hat{M}_1|)$, but is in general higher than the previous scheme as this cost now involves two complete permutations of the matrix coefficients, as well as a reorganization of the block structure of $\hat{M}_1$ which may possibly be computed at runtime. Nevertheless, in situations where memory constraints are significant, partial precomputation schemes of this sort may be preferred.

---

[1] M. Fannes, B. Nachtergaele, and R. Werner, Commun. Math. Phys. **144**, 443 (1992).

[2] S. Ostlund and S. Rommer, Phys. Rev. Lett. **75**, 3537 (1995).

[3] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac, Quantum Inf. Comput. **7**, 401 (2007).

[4] K. G. Wilson, Rev. Mod. Phys. **47**, 773 (1975).

[5] S. R. White, Phys. Rev. Lett. **69**, 2863 (1992).

[6] S. R. White, Phys. Rev. B **48**, 10345 (1993).

[7] U. Schollwöck, Rev. Mod. Phys. **77**, 259 (2005).

[8] I. P. McCulloch, e-print arXiv:0804.2509v1 (unpublished).

[9] G. Vidal, Phys. Rev. Lett. **91**, 147902 (2003).

[10] G. Vidal, Phys. Rev. Lett. **93**, 040502 (2004).

[11] G. Vidal, Phys. Rev. Lett. **98**, 070201 (2007).

[12] A. J. Daley, C. Kollath, U. Schollwöck, and G. Vidal, J. Stat. Mech.: Theor. Exp. (2004) P04005.

[13] S. R. White and A. E. Feiguin, Phys. Rev. Lett. **93**, 076401 (2004).

[14] U. Schollwöck, J. Phys. Soc. Jpn. S **74**, 246 (2005).

[15] Y. Y. Shi, L.-M. Duan, and G. Vidal, Phys. Rev. A **74**, 022320 (2006).

[16] G. Vidal, Phys. Rev. Lett. **99**, 220405 (2007).

[17] G. Vidal, Phys. Rev. Lett. **101**, 110501 (2008).

[18] G. Evenbly and G. Vidal, Phys. Rev. B **79**, 144108 (2009).

[19] V. Giovannetti, S. Montangero, and R. Fazio, Phys. Rev. Lett. **101**, 180503 (2008).

[20] R. N. C. Pfeifer, G. Evenbly, and G. Vidal, Phys. Rev. A **79**, 040301(R) (2009).

[21] G. Vidal, in *Understanding Quantum Phase Transitions*, edited by L. D. Carr (Taylor & Francis, Boca Raton, 2010).

[22] F. Verstraete and J. I. Cirac, e-print arXiv:cond-mat/0407066v1 (unpublished).

[23] G. Sierra and M. A. Martín-Delgado, in *Proceedings of the Workshop on the Exact Renormalization Group, Faro (Portugal)* (World Scientific 1998).

[24] T. Nishino and K. Okunishi, J. Phys. Soc. Jpn. **67**, 3066 (1998).

[25] Y. Nishio, N. Maeshima, A. Gendiar, and T. Nishino, e-print arXiv:cond-mat/0401115v1 (unpublished).

[26] V. Murg, F. Verstraete, and J. I. Cirac, Phys. Rev. A **75**, 033605 (2007).

[27] J. Jordan, R. Orus, G. Vidal, F. Verstraete, and J. I. Cirac, Phys. Rev. Lett. **101**, 250602 (2008).

[28] Z.-C. Gu, M. Levin, and X.-G. Wen, Phys. Rev. B **78**, 205116 (2008).

[29] H. C. Jiang, Z. Y. Weng, and T. Xiang, Phys. Rev. Lett. **101**, 090603 (2008).

[30] Z. Y. Xie, H. C. Jiang, Q. N. Chen, Z. Y. Weng, and T. Xiang, Phys. Rev. Lett. **103**, 160601 (2009).

[31] V. Murg, F. Verstraete, and J. I. Cirac, Phys. Rev. B **79**, 195119 (2009).

[32] L. Tagliacozzo, G. Evenbly, and G. Vidal, Phys. Rev. B **80**, 235127 (2009).

[33] V. Murg, O. Legeza, R. M. Noack, and F. Verstraete, Phys. Rev. B **82**, 205105 (2010).

[34] G. Evenbly and G. Vidal, Phys. Rev. B **81**, 235102 (2010).

[35] G. Evenbly and G. Vidal, New J. Phys. **12**, 025007 (2010).

[36] M. Aguado and G. Vidal, Phys. Rev. Lett. **100**, 070404 (2008).

[37] L. Cincio, J. Dziarmaga, and M. M. Rams, Phys. Rev. Lett. **100**, 240603 (2008).

[38] G. Evenbly and G. Vidal, Phys. Rev. Lett. **102**, 180406 (2009).

[39] R. König, B. W. Reichardt, and G. Vidal, Phys. Rev. B **79**, 195123 (2009).

[40] G. Evenbly and G. Vidal, Phys. Rev. Lett. **104**, 187203 (2010).

[41] P. Corboz, G. Evenbly, F. Verstraete, and G. Vidal, Phys. Rev. A **81**, 010303(R) (2010).

[42] C. V. Kraus, N. Schuch, F. Verstraete, and J. I. Cirac, Phys. Rev. A **81**, 052338 (2010).

[43] C. Pineda, T. Barthel, and J. Eisert, Phys. Rev. A **81**, 050303(R) (2010).

[44] P. Corboz and G. Vidal, Phys. Rev. B **80**, 165129 (2009).

[45] T. Barthel, C. Pineda, and J. Eisert, Phys. Rev. A **80**, 042333 (2009).

[46] Q.-Q. Shi, S.-H. Li, J.-H. Zhao, and H.-Q. Zhou, e-print arXiv:0907.5520v1 (unpublished).

[47] S.-H. Li, Q.-Q. Shi, and H.-Q. Zhou, e-print arXiv:1001.3343v1 (unpublished).

[48] P. Corboz, R. Orus, B. Bauer, and G. Vidal, Phys. Rev. B **81**, 165104 (2010).

[49] I. Pizorn and F. Verstraete, Phys. Rev. B **81**, 245110 (2010).

[50] Z.-C. Gu, F. Verstraete, and X.-G. Wen, e-print arXiv:1004.2563v1 (unpublished).

[51] J. F. Cornwell, *Group Theory in Physics* (Academic, San Diego, 1997).

[52] S. Singh, R. N. C. Pfeifer, and G. Vidal, Phys. Rev. A **82**, 050301 (2010).

[53] S. Singh and G. Vidal (unpublished).

[54] S. Ramasesha, S. K. Pati, H. R. Krishnamurthy, Z. Shuai, and J. L. Bredas, Phys. Rev. B **54**, 7598 (1996).

[55] G. Sierra and T. Nishino, Nucl. Phys. B **495**, 505 (1997).

[56] W. Tatsuaki, Phys. Rev. E **61**, 3199 (2000).

[57] I. P. McCulloch and M. Gulacsi, Europhys. Lett. **57**, 852 (2002).

[58] S. Bergkvist, I. P. McCulloch, and A. Rosengren, Phys. Rev. A **74**, 053419 (2006).

[59] S. Pittel and N. Sandulescu, Phys. Rev. C **73**, 014301 (2006).

[60] I. McCulloch, J. Stat. Mech.: Theor. Exp. (**2007**) P10014.

[61] D. Perez-Garcia, M. M. Wolf, M. Sanz, F. Verstraete, and J. I. Cirac, Phys. Rev. Lett. **100**, 167202 (2008).

[62] M. Sanz, M. M. Wolf, D. Perez-Garcia, and J. I. Cirac, Phys. Rev. A **79**, 042308 (2009).

[63] A. J. Daley, S. R. Clark, D. Jaksch, and P. Zoller, Phys. Rev. A **72**, 043618 (2005).

[64] I. Danshita, J. E. Williams, C. A. R. Sá de Melo, and C. W. Clark, Phys. Rev. A **76**, 043606 (2007).

[65] D. Muth, B. Schmidt, and M. Fleischhauer, New J. Phys. **12**, 083065 (2010).

[66] R. V. Mishmash, I. Danshita, C. W. Clark, and L. D. Carr, Phys. Rev. A **80**, 053612 (2009).

[67] S. Singh, H.-Q. Zhou, and G. Vidal, New J. Phys. **12**, 033029 (2010).

[68] Z. Cai, L. Wang, X. C. Xie, and Y. Wang, Phys. Rev. A **81**, 043602 (2010).

[69] D. Perez-Garcia, M. Sanz, C. E. Gonzalez-Guillen, M. M. Wolf, and J. I. Cirac, New J. Phys. **12**, 025010 (2010).

[70] H. H. Zhao, Z. Y. Xie, Q. N. Chen, Z. C. Wei, J. W. Cai, and T. Xiang, Phys. Rev. B **81**, 174411 (2010).

[71] N. Schuch, I. Cirac, and D. Perez-Garcia, Ann. Phys. **325**, 2153 (2010).

[72] B. Swingle and X.-G. Wen, e-print arXiv:1001.4517v1 (unpublished).

[73] X. Chen, B. Zeng, Z.-C. Gu, I. L. Chuang, and X.-G. Wen, Phys. Rev. B **82**, 165119 (2010).

[74] L. Tagliacozzo and G. Vidal, e-print arXiv:1007.4145v1 (unpublished).

[75] The fundamental theorem of Abelian groups states that every finite Abelian group may be expressed as a direct sum of cyclic subgroups of prime-power order.