Tensor networks contraction and the belief propagation algorithm

R. Alkabetz and I. Arad
Department of Physics, Technion, 3200003 Haifa, Israel

(Received 28 August 2020; revised 11 March 2021; accepted 12 March 2021; published 27 April 2021)

Belief propagation is a well-studied message-passing algorithm that runs over graphical models and can be used for approximate inference and approximation of local marginals. The resulting approximations are equivalent to the Bethe-Peierls approximation of statistical mechanics. Here, we show how this algorithm can be adapted to the world of projected-entangled-pair-state tensor networks and used as an approximate contraction scheme. We further show that the resultant approximation is equivalent to the "mean field" approximation that is used in the simple-update algorithm, thereby showing that the latter is essentially the Bethe-Peierls approximation. This shows that one of the simplest approximate contraction algorithms for tensor networks is equivalent to one of the simplest schemes for approximating marginals in graphical models in general and paves the way for using improvements of belief propagation as tensor networks algorithms.

DOI: 10.1103/PhysRevResearch.3.023073

I. INTRODUCTION

There is a natural connection between classical probabilistic systems of many random variables and quantum many-body systems. In both cases the description of a generic state of a system requires an exponential number of parameters in the size of the system or the number of physical units that compose it. For example, a general probability distribution over n bits requires the specification of $2^n - 1$ non-negative numbers, while a full description of a quantum state over n qubits requires $2^n - 1$ complex numbers. However, in both cases, states that are relevant to us are often subject to many local constraints, which, in turn, may lead to a succinct description of the system. A good example is tensor networks (TNs) [1], where the 2^n coefficients of a quantum state are given by the contraction of a set of local tensors. As a probabilistic analog, consider *graphical models* [2–4], in which a multivariate probability distribution is given by a product of local factors. Tensor networks and graphical models are therefore two frameworks that provide a compact description of the state of the system, which in principle can be used to simulate it.

Both frameworks also face similar challenges. In both cases, calculating the expectation value of a local observable can be an NP-hard problem [5,6], as it (at least naively) involves summation over an exponential number of terms. In addition, when the underlying graph that describes the model is a tree, this can be done efficiently using dynamical programming (via the sum-product algorithm [2] for graphical models or directly by the results of Ref. [7] for tensor networks); however, when there are loops, the problem becomes hard, and one usually resorts to approximations. In the world of tensor networks this is known as the problem of approximate contraction, whereas in the world of graphical models this is known as the problem of approximated inference and local marginals.

Over the years many different algorithms and techniques have been suggested to address this problem for both frameworks. Some of them have been adopted and adjusted to the other framework. For example, the corner-transfer matrix renormalization group (CTMRG) method [8,9] for approximate tensor network contraction has its roots in Baxter's work in statistical mechanics [10,11], as well as ideas of using Monte Carlo sampling for TN contraction [12–14]. From the other side, the tensor renormalization group (TRG) algorithm for the contraction of tensor networks can be used for highly accurate approximations of classical statistic mechanical quantities such as partition functions and magnetization [15] (see also Ref. [16]).

In this paper we show how an important class of inference and marginalization algorithms for graphical models, called belief propagation (BP), can be adapted and used for approximate tensor network contraction. This idea was suggested in Ref. [17], in the context of a general mapping between tensor networks and graphical models. Here, by using a slightly different mapping, we show how this approximation is in fact equivalent to the basic contraction approximation that is at the heart of the simple-update algorithm of tensor networks [18,19]. As we discuss later, since the underlying approximation in the BP algorithm is the Bethe-Peierls approximation, our results imply that this type of approximation is also at the center of the simple-update method. It also motivates the study of various improvements of the BP algorithms as potential tensor network contraction algorithms.

II. A BP ALGORITHM FOR TENSOR NETWORKS

Belief propagation (BP) [20] is a statistical inference algorithm on graphical models that can be used to approximate their marginals [2–4]. It is also

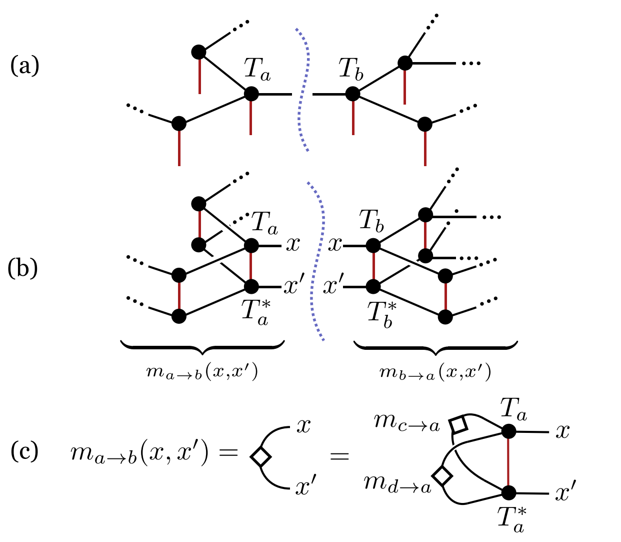


FIG. 1. BP messages on a tree PEPS TN. (a) A local patch of a PEPS defined on a tree. The (a, b) edge defines a bipartition of the system into two branches. (b) The TN that corresponds to $\langle\psi|\psi\rangle$. Also here, the (a, b) edge defines two branches. The tensors that result from the contraction of each branch are the messages $m_{a\to b}(x, x')$ and $m_{b\to a}(x, x')$. (c) The messages satisfy recursion relations that are used to define that BP eqations.

known as the *sum-product algorithm* in the context of coding theory [21] and can also be viewed as an iterative way to solve the Bethe-Peierls equations of statistical physics [22,23]. In what follows, we present a BP variant on a projected-entangled-pair-state (PEPS) tensor network. For an alternative approach, which first maps the PEPS to a graphical model and then uses the BP on that graphical model, please see Appendix A.

We now describe the BP algorithm on PEPS tensor networks. We assume the reader is familiar with the basic notions of tensor networks. For an excellent introduction to the subject we recommend Ref. [1], as well as more recent texts such as Ref. [24]. We consider a PEPS $|\psi\rangle$, in which the physical spins sit on the vertices (nodes) of some graph $G = (V, E)$ [Fig. 1(a)]. Each node $a \in V$ is associated with a tensor T_a that has one physical index (leg) of bond dimension d and a "virtual leg" of dimension D for each adjacent edge. Virtual legs of the same edge in G are contracted together.

We now look at the double-layer TN that corresponds to the scalar $\|\psi\|^2 = \langle\psi|\psi\rangle$ [Fig. 1(b)]. When $G = (V, E)$ is a tree [such as a matrix-product state (MPS), for example], it can be contracted efficiently using dynamical programming. One way to perform it is as follows. Given two incident nodes a, b, the edge that connects them divides the system into two parts; see Figs. 1(a) and 1(b). We define the "message" $m_{a\to b}(x, x')$ to be the tensor that results from contracting the branch of $\langle\psi|\psi\rangle$ that is connected to a, where (x, x') are the indices of the open ket-bra edges connected to node a. Similarly, the message $m_{b\to a}(x, x')$ is related to the contraction over the branch of b. See Figs. 1(a) and 1(b). If we consider them as matrices of the indices (x, x'), they are positive semidefinite, due to the fact that they are a result of a contraction of a branch with its complex conjugate. Crucially, these messages satisfy

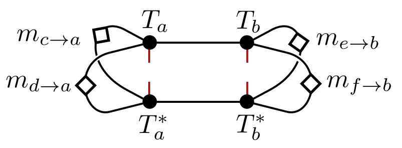


FIG. 2. In a tree, the converged BP messages can be used to calculate the local RDMs. In this example, a two-local RDM is shown. The same formulas are used to approximate the local RDMs also when the underlying PEPS is not a tree.

a recursive relation. If N_a is the set of nodes that are incident to a, then the tensor $m_{a\to b}$ is given by the contraction

$$m_{a\to b} = \mathrm{Tr}\left(T_a T_a^* \prod_{a'\in N_a\setminus\{b\}} m_{a'\to a}\right), \tag{1}$$

where $\mathrm{Tr}(\cdot)$ denotes contraction of joint indices. For example, if b, c, d are incident to a, then the message $m_{a\to b}(x, x')$ is given in terms of the messages $m_{c\to a}(x, x')$ and $m_{d\to a}(x, x')$, as shown in Fig. 1(c).

In principle, we can pick any node which is not a leaf, define it as a root, and use Eq. (1) to calculate the messages from the leaves to the root. Using the messages that lead to the root, we can calculate $\|\psi\|^2$. This calculation can also be done differently. Instead of forcing a particular causality order between the messages, we can try to solve Eq. (1) for *all* messages simultaneously. This can be done by solving Eq. (1) iteratively: Starting from a set of random positive semidefinite (PSD) messages $\{m^{(0)}_{a\to b}(x, x')\}$ for all incident nodes a, b, we define the set of messages at step $t + 1$ using the messages of step t:

$$m^{(t+1)}_{a\to b} \overset{\mathrm{def}}{=} \mathrm{Tr}\left(T_a T_a^* \prod_{a'\in N_a\setminus\{b\}} m^{(t)}_{a'\to a}\right). \tag{2}$$

Equation (2) is the BP equation for PEPS tensor networks. It is a natural extension of the BP equations of graphical models [2–4]. The equation also guarantees that if the messages at t are PSD when viewed as a matrix whose (x, x') element is $m_{a\to b}(x, x')$, then so would be the messages at $t + 1$. The fixed point of this iterative process will solve Eq. (1) and give us all the $m_{a\to b}(x, x')$ messages. It is a well-known fact that the BP iterations on tree graphical models have a unique fixed point to which they converge in a linear number of steps [25]. The same arguments easily generalize also to our case. Once we have the messages, we can use the fact that they are contractions over branches and use them to calculate local reduced density matrices (RDMs). For example, the calculation of two-local RDMs is shown in Fig. 2. The PSD property of the messages guarantees that the resultant RDMs are also PSD.

Thus far, the BP equations (2) might seem no more than an elegant method for contracting *tree* PEPS. Things become interesting when we consider graphs with loops. In such a case, the messages can no longer be defined as the contraction of branches, since an edge in the graph no longer partitions it into two distinct branches. Yet we can still define them as solutions to Eq. (1) and try to find them iteratively using Eq. (2). If the iterations converge to a fixed point, we can use the messages to estimate local marginals, using the *same* expressions as we did in the case of trees, which are illustrated in Fig. 2. This procedure is called *loopy BP*. Evidently, this is an uncontrolled approximation. In fact, there is no guarantee that the BP iterations will converge to a fixed point or that there is a unique fixed point. Nevertheless, in the world of graphical models, the BP method often provides surprisingly good results, in particular when the graph looks locally like a tree and there are no long-range correlations. A famous example consists of problems related to the decoding of error correction codes, in which BP performs extremely well [26]. As we shall see, also in the world of tensor networks, loopy BP often performs well on problems with short-range entanglement.

While a general theory to explain the performance of the BP algorithm is still lacking, there are some partial results in this direction. An important result is due to Yedidia *et al.* [27], who established the correspondence between fixed points of the BP algorithm and the Bethe-Peierls approximation [22]. The Bethe-Peierls approximation is an approximation scheme for classical statistical mechanics, in which one treats the system as if it is defined on a tree (a Bethe lattice). This assumption implies that the Gibbs distribution of the system, as well as the free-energy functional, can be written as functions of the local marginals. When the actual interaction graph of the system is not a tree, this is an uncontrolled approximation. Nevertheless, also in these cases, one can take the Bethe free-energy functional and look for a locally consistent set of marginals that minimizes it. This procedure often gives surprisingly good approximations. Yedidia *et al.* [27] showed that there is a one-to-one correspondence between stationary points of the Bethe free energy and fixed points of the BP equations. The marginals obtained from the messages at a fixed point minimize the Bethe free energy, and conversely, from the marginals at the stationary point one can derive fixed-point messages of BP.

As we show in Appendix A 2, this result naturally generalizes to our case. The idea is that our BP algorithm for PEPS is the usual BP algorithm that is applied to a graphical model with complex entries, obtained from the tensor network of $\langle\psi|\psi\rangle$. Such graphical models were studied in Ref. [28], under the name "double-edge factor graph" (DEFG), where it was argued that the BP algorithm corresponds, as in the usual case, to the stationary points of the Bethe free energy.

At this point it seems tempting to benchmark the BP algorithm for PEPS contraction and compare it to other methods. Indeed, we first used the BP algorithm as a contraction subroutine in an imaginary time evolution algorithm on various two-dimensional (2D) models and compared it with the performance of the simple-update method [29]. Surprisingly, the final energies of both algorithms were suspiciously close to each other. As we show next, this can be explained theoretically; the mean field approximation at the heart of the simple-update method and the BP algorithm are equivalent.

III. THE SIMPLE-UPDATE METHOD

The simple-update method [18] is a direct generalization of the time-evolving block decimation (TEBD) [30–32] algorithm for 1D real and imaginary time evolution to

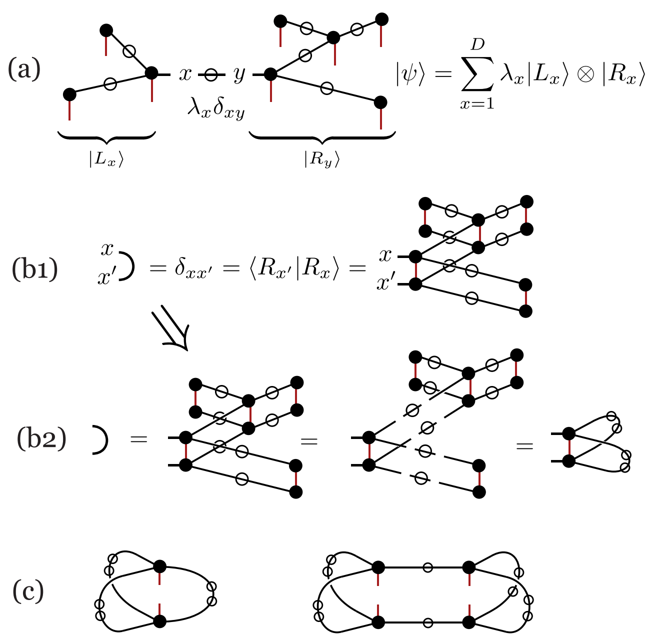

(a)
x
y
$\lambda_x\delta_{xy}$
$|L_x\rangle$
$|R_y\rangle$
$|\psi\rangle = \sum_{x=1}^{D} \lambda_x |L_x\rangle \otimes |R_x\rangle$
(b1)
x
x'
$= \delta_{xx'} = \langle R_{x'}|R_x\rangle =$
(b2)
=
=
=
(c)

FIG. 3. The properties of a canonical representation of a tree PEPS. (a) An example of a canonical representation of a tree PEPS and its relation to the Schmidt decomposition. The empty circles are diagonal tensors that correspond to the Schmidt weights λ. (b1) The orthonormality of the Schmidt bases implies a simple formula for the contraction of the left and right branches and (b2) a *local* canonical condition on the PEPS tensors. (c) A local expression for the reduced density matrices.

higher dimensions. It calculates the dynamics of many-body spin systems that sit on a lattice and are described by a (quasi)canonical PEPS tensor network. The method is efficient and numerically stable but often results in poor accuracy due to its oversimplified representation of local environments.

At its core lies a quasicanonical form of the PEPS that allows a crude approximation of local TN environments, often referred to as the *mean field approximation*. In this form, in addition to the local tensors of the PEPS, there are also diagonal λ tensors in the middle of every edge, shown as empty circles in Fig. 3. When the PEPS is in the shape of a tree, the form becomes truly canonical; every edge corresponds to a bipartition of the system, and its λ tensors become the Schmidt weights of the corresponding bipartition, as shown in Fig. 3(a).

When the underlying graph has loops, the canonical form is no longer well defined; removing an edge from the graph no longer divides it into two parts, and so it cannot be associated with a Schmidt decomposition between two branches. Nevertheless, we can still define a PEPS to be *quasicanonical* if it satisfies the *local* conditions of Fig. 3(b2). In such a case, the expression for local RDMs [e.g., Fig. 3(c)] is no longer exact. Yet, when the graph looks locally like a tree, or when the quantum state has only short-range correlations, this approximation is often reasonable.

In the simple-update method, one evolves the system in real or imaginary time using the Trotter-Suzuki decomposition, while striving to keep the quasicanonical form along the real or imaginary time evolution. This is done using local singular value decompositions (SVDs) on the tensors. See, for example, Ref. [19] and Appendix B. Importantly, by using a trivial time evolution (i.e., setting the Hamiltonian to zero), one can use these SVDs to iteratively drive any PEPS to quasicanonical form. See Appendix B for details. We call this procedure the trivial-simple-update algorithm (*trivial-SU algorithm*) and note that it can also be viewed as an algorithm for approximate contraction of the PEPS, because once we have the quasicanonical form, we can use it to approximate the local RDMs, as in Fig. 3(c). Importantly, while the underlying PEPS changes in this process, it still represents exactly the same quantum state. The trivial-SU algorithm has already been used previously under different names such as the quasiorthogonalization in Appendix B of Ref. [33] or the superorthogonalization in Ref. [34]. Given that the trivial-SU algorithm does not change the overall state of the system, it can be seen also as a special case of a holographic transformation on a normal factor graph (NRF). NRFs are an equivalent framework to tensor networks for representing multivariate functions in a graphical notation. For a detailed explanation of NRF and holographic transformations, see Ref. [35].

IV. BP-SU EQUIVALENCE

The trivial-SU algorithm and the BP algorithm for TN are two different algorithms for approximate contraction of PEPS. They originate from two very different places: The trivial-SU algorithm is a natural algorithm for TN, which relies on the Schmidt decomposition and SVD, whereas the BP algorithm is a message-passing algorithm for graphical models that originated from inference problems and the Bethe-Peierls approximation. It might therefore come as a surprise that these two algorithms are equivalent. In hindsight, this could have been anticipated, as both are exact on trees. We prove the following theorem.

Theorem 1. Every trivial-SU fixed point corresponds to a BP fixed point such that the local RDMs computed in both methods are identical.

As a simple corollary, we conclude that if the trivial-SU equations and the BP equations have a unique fixed point, both algorithms will yield the same RDMs. In light of this equivalence, the success of the imaginary time SU algorithm for many models (see, for example, models analyzed in Ref. [19]) is another example of the success of the Bethe-Peierls approximation.

To prove Theorem 1, we note that in the BP algorithm the TN remains fixed, while the BP messages evolve to a fixed point. In the trivial-SU algorithm, there are no messages, but the local tensors that make up the TN evolve until they converge to a quasicanonical fixed point, without changing the underlying quantum state. In both cases, the evolution is done via local steps. Our proof uses two lemmas.

Lemma 1. Let $\mathcal{T}, \mathcal{T}'$ be two tensor networks that represent the same state $|\psi\rangle$, such that $\mathcal{T}'$ is obtained from $\mathcal{T}$ using a single trivial-SU step on tensors T_a, T_b and the λ weight between them. Then every BP fixed point of $\mathcal{T}$ has a corresponding fixed point of $\mathcal{T}'$ with the same RDMs, and vice versa.

The idea of the proof is to show that the fixed-point messages of the new TN can be constructed from the fixed-point messages of the old TN, except for the local place of change, where the messages are adapted to fit the new tensors. The full proof of the lemma is given in Appendix C 1.

Using this lemma repeatably along the trivial-SU iterations, we conclude that the BP fixed points of an initial TN are equivalent to those of its quasicanonical representation. To finish the proof, we show that the BP fixed point of a quasicanonical PEPS yields the same RDMs as the λ weights do:

Lemma 2. Given a TN in a quasicanonical form (i.e., a fixed point of the trivial-SU algorithm), it has a BP fixed point that gives the same RDM estimates as those of the quasicanonical form based on the λ weights.

The idea of the proof is that after "swallowing" a $\sqrt{\lambda}$ of each λ tensor in its two adjacent T_a, T_b tensors, we reach a PEPS for which the messages $m_{a\to b}(x,x') = \lambda_x\delta_{x,x'}$ are a BP fixed point. The full proof is found in Appendix C 2.

Numerical Tests

As we saw, both BP and trivial-SU algorithms produce the same RDM approximations. In addition, the asymptotic complexity of a single step in both algorithms is $O(dD^k)$, where k is the local degree of a vertex. What is less clear is the number of iterations needed for both algorithms to converge. We compared these numbers on two types of PEPS on finite square grids: a random PEPS with complex entries, and an approximate ground state of the antiferromagnetic Heisenberg model on a square lattice with random, nearest-neighbor coupling. Our numerics indicate that the convergence times of both algorithms are similar. While for the random PEPS, the BP seemed slightly faster ($T_{\mathrm{BP}}/T_{\mathrm{tSU}} \simeq 0.7$), for the antiferromagnetic Heisenberg model, the trivial-SU algorithms seemed to converge faster ($T_{\mathrm{BP}}/T_{\mathrm{tSU}} \simeq 1.5$). Full details of the numerical procedure and the results can be found in Appendix D.

Finally, we stress that in all our numerical tests, the marginals produced by both methods (the trivial-SU and the BP algorithms) were identical up to the convergence ϵ of the algorithms (which was typically 10^{-6}).

V. DISCUSSION

In this paper, we have defined the belief propagation method for PEPS contraction, which can be viewed as the ordinary BP method applied for double-edge factor graphs (DEFGs) [28] that are derived from the PEPS tensor network. Just as in ordinary graphical models, the fixed points of the BP iterations correspond to stationary points of the Bethe free energy, which is defined for the underlying PEPS TN. We have shown that the BP algorithm is equivalent to the trivial-SU algorithm on PEPS tensor networks, which leads to a quasicanonical form. This correspondence has some interesting implications. First, since the fixed points of the imaginary time SU algorithm are quasicanonical PEPS tensor networks, our result implies that their SU approximate environments correspond to a Bethe-Peierls approximation. The success of the SU algorithm can therefore be seen as another example of the power of the Bethe-Peierls approximation. Indeed, just like the BP algorithm, the SU algorithm often gives surprisingly good results, even when the underlying graph has many short-range loops and is very different from a tree, as, for example, in the case of square lattice graphs [18,19]. Second, it shows that one of the simplest algorithms for approximating marginals in the world of graphical models is equivalent to one of the simplest approximate contraction algorithms in the world of tensor networks. Therefore it would be interesting to "import" other, more sophisticated algorithms for marginal approximations to the world of tensor networks. A natural candidate is the generalized belief propagation (GBP) algorithm [27], which produces much better approximations, at the price of a higher computational cost. It generalizes the BP algorithm by considering messages from larger regions in the graph, corresponding to Kikuchi's cluster variation method [36,37]. It would be interesting to compare the performance of this algorithm, as well as other BP improvements [38–42], when acting on TNs with that of more accurate contraction algorithms, such as the corner-transfer matrix renormalization group (CTMRG) method [8,9], the tensor renormalization group (TRG) algorithm [15,43], and the boundary MPS (bMPS) technique [44], to name a few.

From a theoretical prospective, it would be interesting to better understand the physical and mathematical role of the complex Bethe free energy that we have derived. In particular, we know that for tree tensor networks, it is related to the Schmidt decomposition. Can we somehow relate it to the underlying entanglement structure also when the underlying graph has loops? Another interesting question is whether the BP equations can be used to analytically analyze models for which the ground state is a known PEPS, such as the Affleck-Kennedy-Lieb-Tasaki (AKLT) model. Finally, we note that unlike the trivial-SU algorithm, our BP scheme easily generalizes to mixed states described by projected-entangled-pair-operator (PEPO) tensor networks, for which there is no natural Schmidt decomposition.

ACKNOWLEDGMENTS

The authors thank Eyal Bairy, Raz Firanko, Roman Orús, and Yosi Avron for their help with the manuscript and many useful suggestions. I.A. acknowledges the support of the Israel Science Foundation (ISF) under Individual Research Grants No. 1778/17 and No. 2074/19.

APPENDIX A: GRAPHICAL MODELS, BELIEF PROPAGATION, AND THE MAPPING OF PEPS TENSOR NETWORKS AND DOUBLE-EDGE FACTOR GRAPHS

In this Appendix we give a very brief background on the subject of graphical models and the belief propagation algorithm and then sketch a mapping between the PEPS tensor networks and a particular type of graphical models called double-edge factor graphs. Together, this will show that our BP algorithm for PEPS is essentially the usual BP algorithm applied to the mapped double-edge factor graphs, and the converged messages correspond to a Bethe-Peierls approximation.

1. Graphical models and belief propagation

Graphical models are a powerful tool for modeling multivariate probability distributions. They provide a succinct description of the statistical dependence of a set of random variables using graphs and are used in fields such as bioinformatics, communication theory, statistical physics, combinatorial optimization, signal and image processing, and statistical machine learning, to name a few. In this section we give a brief background on this tool and the BP algorithm. For a thorough review, we refer the reader to Refs. [2–4].

Roughly speaking, there are three main families of graphical models: Bayesian networks, Markov random fields (MRFs), and factor graph graphical models. As the latter family supersedes the first two, we will concentrate on it.

A factor graph graphical model is a succinct description multivariate function $f(x_1,\dots,x_n)$, given as a product of functions over subsets of these variables:

$$f(x_1,\dots,x_n)=\prod_{a\in\mathcal{F}} f_a(\boldsymbol{x}_a).$$

Here, $f(x_1,\dots,x_n)$ is called a *global function*, and the $\{f_a\}$ functions are called *factors*. The factors are defined over a subset of variables $\boldsymbol{x}_a=(x_{i_1},x_{i_2},\dots,x_{i_k})$, where typically we are interested in the cases where $k=O(1)$. Finally, we use $\mathcal{F}$ to denote the set of all factors and $\boldsymbol{x}=(x_1,\dots,x_n)$ as a shorthand notation for the set of all variables.

When the factors are all non-negative functions, the global function $f(x_1,\dots,x_n)$ also becomes non-negative and can be used to model multivariate *probability distributions*

$$P(\boldsymbol{x})=\frac{1}{Z}\prod_{a\in\mathcal{F}} f_a(\boldsymbol{x}_a),$$

where $1/Z$ is a normalization factor given by $Z=\sum_{\boldsymbol{x}}\prod_{a\in\mathcal{F}} f_a(\boldsymbol{x}_a)$. In this paper, we mainly focus on such probabilistic factor graphs, which are a particular type of *probabilistic graphical models*. In Appendix A 2 we also discuss how factor graphs can describe quantum states.

Probabilistic factor graphs capture many natural probability distributions from a wide range of scientific areas such as physics, economics, machine learning, coding theory, etc. For example, in classical statistical mechanics the Gibbs distribution is of that form: Given a local Hamiltonian $H(\boldsymbol{x})=\sum_{a\in\mathcal{F}} h_a(\boldsymbol{x}_a)$, its Gibbs distribution at inverse temperature β is

$$P(\boldsymbol{x})=\frac{1}{Z}e^{-\beta H(\boldsymbol{x})}=\frac{1}{Z}\prod_{a\in\mathcal{F}} e^{-\beta h_a(\boldsymbol{x}_a)}.$$

In this case, $f_a(\boldsymbol{x}_a)=e^{-\beta h_a(\boldsymbol{x}_a)}$, and Z is the partition function.

There is a convenient graphical way to capture the relation between the various factors, using a so-called *factor graph*$\mathcal{G}=(\mathcal{V},\mathcal{F},\mathcal{E})$. This is a bipartite graph with two types of vertices: $\mathcal{V}=\{x_1,x_2,\dots,x_n\}$ is the set of variables, also called *nodes* or *variable nodes*. The other set of vertices are the factors $\mathcal{F}=\{f_1,f_2,\dots\}$, also called *factor nodes*. $\mathcal{E}$ is the set of edges, where an edge connects the node x_i to the factor f_a if and only if f_a depends on x_i. For example, the factor graph in Fig. 4 corresponds to probability distributions of the form $P(x_1,x_2,x_3)=\frac{1}{Z}f_1(x_1,x_2,x_3)\cdot f_2(x_1,x_2)$.

Given a graphical model, a central task is to calculate marginals of $P(\boldsymbol{x})$ over some small set of random variables. This is needed, for example, for the calculation of local expectation values or for the optimization of the model with respect

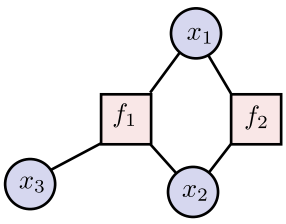


FIG. 4. An example of a factor graph representing the probability distribution $P(x_1, x_2, x_3) = \frac{1}{Z} f_1(x_1, x_2, x_3) \cdot f_2(x_1, x_2)$.

to empirical data. This task is NP-hard in general, involving a summation over an exponential number of configurations [5].

Belief propagation (BP) [2–4,45] is a message-passing algorithm that is designed to approximate such marginals. It is exact on graphical models whose underlying graph is a tree and often gives surprisingly good results on loopy graphs. In these cases, however, it is essentially an uncontrolled heuristic. The BP algorithm is often known by different names in different contexts. In statistical physics, it is known as the "Bethe-Peierls approximation" [22,23], and in coding theory as the "sum-product algorithm" [21]. The name "belief propagation" was coined by Pearl, who used it in the context of Bayesian networks [20,45].

The main objects in the BP algorithm are "messages" between factor and nodes and vice versa. To write them, let us define N_i as the set of factors in which x_i participates, and similarly N_a to be the set of variables of the factor f_a (i.e., adjacent nodes to f_a). A message from a factor f_a to an adjacent node $i \in N_a$ is a non-negative function $m_{a\to i}(x_i)$, and a message from node i to factor a is a non-negative function $m_{i\to a}(x_i)$. The BP algorithm starts by initializing the messages (say, randomly), and then at each step, the messages are updated from the messages of the previous step by the local rules (see also Fig. 5):

$$m^{(t+1)}_{i\to a}(x_i) \overset{\text{def}}{=} \prod_{b\in N_i\setminus\{a\}} m^{(t)}_{b\to i}(x_i), \tag{A1}$$

$$m^{(t+1)}_{a\to i}(x_i) \overset{\text{def}}{=} \sum_{\boldsymbol{x}_a\setminus\{x_i\}} f_a(\boldsymbol{x}_a) \prod_{j\in a\setminus\{i\}} m^{(t)}_{j\to a}(x_j). \tag{A2}$$

If the messages converge to a fixed point, they can be used to estimate the marginal on a subset of nodes with a local treelike structure. For example, the marginal of a single variable x_i is given by

$$P_i(x_i) = \frac{1}{\mathcal{N}} \prod_{a\in N_i} m_{a\to i}(x_i), \tag{A3}$$

where $\mathcal{N}$ is a normalization factor. The marginal over the variables of a factor is given by

$$P_a(\boldsymbol{x}_a) = \frac{1}{\mathcal{N}} f_a(\boldsymbol{x}_a) \prod_{i\in N_a} m_{i\to a}(x_i). \tag{A4}$$

These two expressions are demonstrated in Fig. 6.

For tree graphical models, the BP messages are promised to converge to a unique fixed point in linear time, and formulas (A3) and (A4) give the exact marginals [25].

When the underlying graph has loops, the BP algorithm is called "*loopy BP*," and the formulas for the marginals become, essentially, uncontrolled. Moreover, it is not known how fast the algorithm will converge, if ever, or if it has a unique fixed point. Nevertheless, in many practical cases, loopy BP provides surprisingly good results.

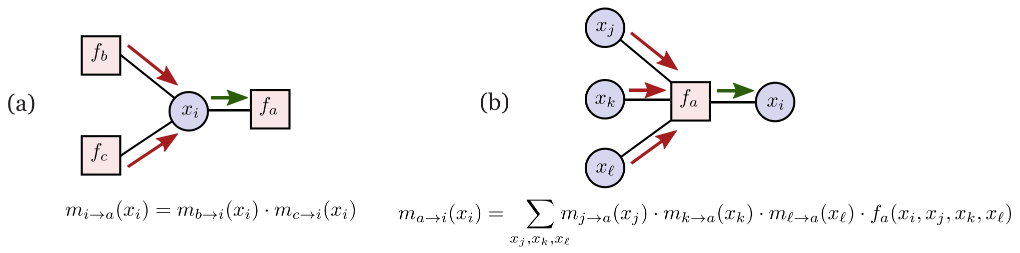
While a general theory to explain the performance of the BP algorithm is still lacking, there are some partial results in this direction. An important result is due to Yedidia *et al.* [27], who highlighted the correspondence between the Bethe-Peierls approximation and fixed points of the BP algorithm, which we now explain briefly. The starting point consists of models defined on tree graphs. A simple observation is that for these models, the global probability distribution can be written in terms of its local marginals:

$$P(\boldsymbol{x}) = \prod_{a\in\mathcal{F}} P_a(\boldsymbol{x}_a) \prod_{i\in\mathcal{V}} [P_i(x_i)]^{1-d_i}, \tag{A5}$$

where $P_a(\boldsymbol{x}_a)$ is the marginal on the nodes adjacent to $a \in \mathcal{F}$ and $P_i(x_i)$ is the marginal of x_i. Finally, $d_i = |N_i|$ is the number of factors that are adjacent to x_i. Using Eq. (A5), we can write the free-energy function in terms of the local marginals:

$$F_{\text{Bethe}} = \sum_a \sum_{\boldsymbol{x}_a} P_a(\boldsymbol{x}_a) \ln \frac{P_a(\boldsymbol{x}_a)}{f_a(\boldsymbol{x}_a)} - \sum_i (d_i - 1) \sum_{x_i} P_i(x_i) \ln P_i(x_i). \tag{A6}$$

The above expression is called the *Bethe free energy*. When the underlying graph is not a tree, the Bethe free energy is still well defined but no longer equals the exact free energy. In such a case, we can use it to approximate the local marginals. We write the Bethe free energy as a function of unknown marginals $\{q_a(\boldsymbol{x}_a), q_i(x_i)\}$, and then we estimate the real marginals $\{P_a(\boldsymbol{x}_a), P_i(x_i)\}$ by finding the $\{q_a(\boldsymbol{x}_a), q_i(x_i)\}$ that *minimize* the Bethe free energy. This procedure is exact on trees, where the Bethe free energy is equal to the exact free energy, but on loopy graphs it is essentially an uncontrolled



(a)
f_b
x_i
f_a
f_c
$m_{i\to a}(x_i) = m_{b\to i}(x_i) \cdot m_{c\to i}(x_i)$
(b)
x_j
x_k
f_a
x_i
x_ℓ
$m_{a\to i}(x_i) = \sum_{x_j,x_k,x_\ell} m_{j\to a}(x_j) \cdot m_{k\to a}(x_k) \cdot m_{\ell\to a}(x_\ell) \cdot f_a(x_i, x_j, x_k, x_\ell)$

FIG. 5. Illustration of the BP equations on a factor graph graphical model: Eqs. (A1) and (A2). In such a case there are two types of messages: (a) nodes to factors [Eq. (A1)] and (b) factors to nodes [Eq. (A2)].

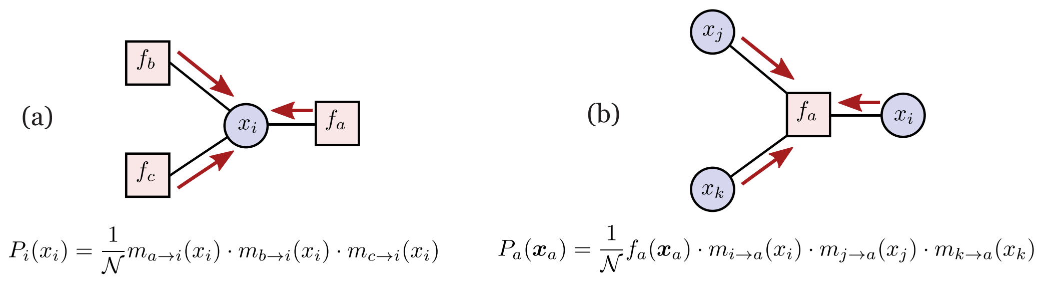

(a)
f_b
x_i
f_a
f_c
$P_i(x_i) = \frac{1}{\mathcal{N}} m_{a\to i}(x_i)\cdot m_{b\to i}(x_i)\cdot m_{c\to i}(x_i)$
(b)
x_j
f_a
x_i
x_k
$P_a(\boldsymbol{x}_a) = \frac{1}{\mathcal{N}} f_a(\boldsymbol{x}_a)\cdot m_{i\to a}(x_i)\cdot m_{j\to a}(x_j)\cdot m_{k\to a}(x_k)$

FIG. 6. Calculating the local marginals from the BP messages by the formulas in Eqs. (A3) and (A4). These formulas give the exact marginals when the underlying graphical model is a tree.

approximation; the resultant $q_a(\boldsymbol{x}_a)$, $q_i(x_i)$, may be far from the exact marginals, and in fact, they might not be marginals of any underlying global distribution. Nevertheless, decades of experience in statistical mechanics have shown that this is often a good approximation that gives better results than simple mean field. In Ref. [27] it was shown that there is a one-to-one connection between the fixed points of the BP equations and the stationary points of the Bethe free energy. The Lagrange multipliers used to minimize the latter become the fixed-point BP messages, and the local marginals coincide. This connection between a message-passing inference algorithm and a variational approach gave rise to a plethora of other message-passing algorithms, such as generalized belief propagation (GBP), which are based on more sophisticated free energies, such as the Kikuchi's cluster variation method [36].

2. Mapping a PEPS tensor network to a graphical model

In this section we present a mapping that takes a PEPS TN to a graphical model. Relations and dualities between graphical models and tensor networks have been studied over the years by several authors [17,46,47]. Our approach shares some similarities with these works but, in particular, builds on the double-edge factor graph (DEFG) formalism of Ref. [28], which by itself uses ideas introduced in Ref. [48] of representing quantum states and quantum processes using factor graphs with complex entries. This allows us to transform a tree tensor network into a tree graphical model, and it also has the desirable property of messages being positive semidefinite matrices.

The mapping between PEPS and DEFG is illustrated in Fig. 7. Let $|\psi\rangle$ be the many-body quantum state that is described by our TN, and consider the tensor network corresponding to $\langle\psi|\psi\rangle$, in which we clump every edge in $|\psi\rangle$ with its equivalent edge in $\langle\psi|$ [see Fig. 7(b)]. We call such pairs of edges "double edges." They run over D^2 values of the double indices (x, x') of the ket and the bra TN. We map this TN into a graphical model as follows:

(1) We associate every double edge with a node so that its double indices (x_i, x_i') now become a single variable in the graphical model. We denote this pair by a single variable $z_i = (x_i, x_i')$ and notice that it runs over D^2 discrete values.

(2) We associate the contraction of every pair T_a, T_a^* of bra-ket local tensors along their physical leg with a factor. See Figs. 7(b) and 7(c). Specifically, let $T^{\mu}_{a;x_1,\dots,x_k}$ be the PEPS tensor at node a, with μ being the physical leg; then the resultant factor is given by

$$f_a[z_1,\dots,z_k] \overset{\text{def}}{=} f_a[(x_1,x_1'),\dots,(x_k,x_k')] \overset{\text{def}}{=} \sum_{\mu=1}^{d} T^{\mu}_{a;x_1,\dots,x_k}\cdot \left(T^{\mu}_{a;x_1',\dots,x_k'}\right)^*. \tag{A7}$$

See Fig. 8. As in the body of the paper, we write $\boldsymbol{x}_a = (x_1,\dots,x_k)$, $\boldsymbol{x}_a' = (x_1',\dots,x_k')$, and $\boldsymbol{z}_a = (z_1,\dots,z_k)$ for the variables of the factor a. With this notation, we may write $f_a(\boldsymbol{z}_a) = f_a(\boldsymbol{x}_a,\boldsymbol{x}_a')$. Definition (A7) immediately implies that as a matrix, $f_a(\boldsymbol{x}_a,\boldsymbol{x}_a')$ is positive semidefinite.

(3) Graphically, variable nodes are denoted by circles, and factors are denoted by squares. Adjacent variables and factors are connected by double lines (edges) that correspond to the double variable $z_i = (x_i, x_i')$ that they represent. See Fig. 7.

With these definitions, the resultant graphical model is called a DEFG and describes the function

$$P(\boldsymbol{z}) \overset{\text{def}}{=} P(z_1,\dots,z_n) = \frac{1}{Z}\prod_a f_a(\boldsymbol{z}_a),$$
$$Z \overset{\text{def}}{=} \sum_{\boldsymbol{z}}\prod_a f_a(\boldsymbol{z}_a) = \langle\psi|\psi\rangle. \tag{A8}$$

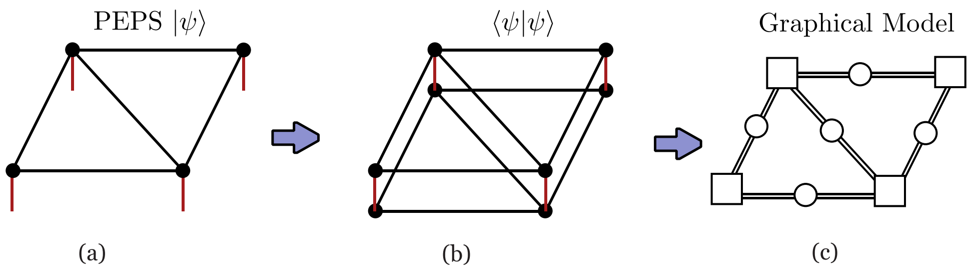


FIG. 7. (a)–(c) Mapping a tensor network to a graphical model of type double-edge factor graph.

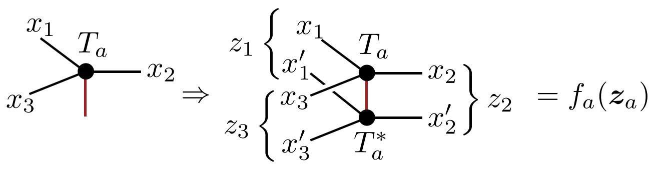


FIG. 8. Defining the complex $f_a(z_a)$ factors from the PEPS tensors T_a, via $f_a(z_a) \overset{\text{def}}{=} \text{Tr}(T_a T_a^*)$. $z_i \overset{\text{def}}{=} (x_i, x'_i)$ that originate from the ket and the bra of the $\langle\psi|\psi\rangle$ TN.

Writing $P(z)$ as $P(\boldsymbol{x}, \boldsymbol{x}')$, the positive semidefiniteness of the individual $f_a(\boldsymbol{x}_a, \boldsymbol{x}'_a)$ implies that $P(\boldsymbol{x}, \boldsymbol{x}')$ is also a positive semidefinite function. We can therefore interpret it as the density matrix of some fictitious quantum states that "lives on the edges of the PEPS," although it has a nonconventional normalization because $\text{Tr}P = \sum_{\boldsymbol{x},\boldsymbol{x}} P(\boldsymbol{x}, \boldsymbol{x})$ is not necessarily equal to 1 [instead, it is $\sum_{\boldsymbol{x},\boldsymbol{x}'} P(\boldsymbol{x}, \boldsymbol{x}') = 1$].

Once the factor graphical model is defined, we can run the BP iterations on it,

$$m^{(t+1)}_{i\to a}(z_i) \overset{\text{def}}{=} \prod_{b\in N_i\setminus\{a\}} m^{(t)}_{b\to i}(z_i), \tag{A9}$$

$$m^{(t+1)}_{a\to i}(z_i) \overset{\text{def}}{=} \sum_{z_a\setminus\{z_i\}} f_a(z_a) \prod_{j\in a\setminus\{i\}} m^{(t)}_{j\to a}(z_j), \tag{A10}$$

which are simply the usual BP equations (A1) and (A2) with x_i replaced by the double-edge variable z_i. It is easy to see that these equations are equivalent to Eq. (2) in the main text by noting that every node i is adjacent to exactly two factors a, b (because it corresponds to an edge in the PEPS connecting two vertices), and therefore by Eq. (A9),

$$m^{(t+1)}_{i\to b}(z_i) = m^{(t)}_{a\to i}(z_i),$$

which we identify with $m^{(t+1)}_{a\to b}$ from Eq. (2). Moreover, the summation $\sum_{z_a\setminus\{z_i\}}$ in Eq. (A10) is exactly the contraction of the virtual legs in Eq. (2) and Fig. 1(c). Finally, note that as $f_a(\boldsymbol{x}_a, \boldsymbol{x}'_a)$ are positive semidefinite, Eqs. (A9) and (A10) imply that if the messages at time t are positive semidefinite, then so are the messages at $t+1$.

The above discussion shows that as in the ordinary graphical models, here also fixed points of the BP iterations are solving a Bethe-Peierls type of approximation. In particular, defining the local "marginals"

$$P_a(z_a) \overset{\text{def}}{=} \sum_{z\setminus z_a} P(z), \qquad P_i(z_i) \overset{\text{def}}{=} \sum_{z\setminus\{z_i\}} P(z),$$

we can write a complex Bethe free energy

$$F_{\text{Bethe}} = \sum_a \sum_{z_a} P_a(z_a) \ln \frac{P_a(z_a)}{f_a(z_a)} - \sum_i (d_i - 1) \sum_{z_i} P_i(z_i) \ln P_i(z_i), \qquad z_i = (x_i, x'_i), \tag{A11}$$

which is defined by first choosing a specific branch of the logarithmic function. Note that in this case, $d_i = |N_i| = 2$ because there are always exactly two adjacent factors to each variable z_i, and so

$$F_{\text{Bethe}} = \sum_a \sum_{z_a} P_a(z_a) \ln \frac{P_a(z_a)}{f_a(z_a)} - \sum_i \sum_{z_i} P_i(z_i) \ln P_i(z_i). \tag{A12}$$

It is not very hard to show that even though $P_a(z)$, $P_i(z_i)$, $f_a(z_a)$ might take complex values, F_{Bethe} must be real. In Ref. [28] it was argued that also in this case, fixed points of the BP iterations correspond to stationary points of the above functional. We note, however, that unlike the ordinary case, we currently see no reason why the complex Bethe free energy should be positive—even though $P(z)$ is positive semidefinite. Interestingly, in all of our numerics, it was positive.

APPENDIX B: THE SIMPLE-UPDATE METHOD

In this Appendix we give a brief review of the simple-update (SU) method and the associated trivial-SU algorithm. For full details, we refer the reader to Refs. [19,49].

In the simple-update algorithm, one usually starts from a quasicanonical PEPS and then applies local gates that perform real or imaginary time evolution according to the Trotter-Suzuki decomposition. For example, in the imaginary time evolution, if the Hamiltonian interaction term between the neighboring sites a, b is h_{ab}, the operator $U_{ab} = e^{-\delta\tau h_{ab}}$ will be applied, where $\delta\tau$ is a small Trotter-Suzuki time step. Once U_{ab} is applied, a local SVD is performed, as shown in Figs. 9(b)–9(f), which guarantees that (i) the resultant tensor network can be reshaped into a local PEPS with T_a, λ, T_b replaced by T'_a, λ', T'_b, (ii) a truncation is performed so that the bond dimension of new tensors does not increase, and (iii) *some* of the local canonical conditions are (approximately) satisfied [50]—see Fig. 9(g).

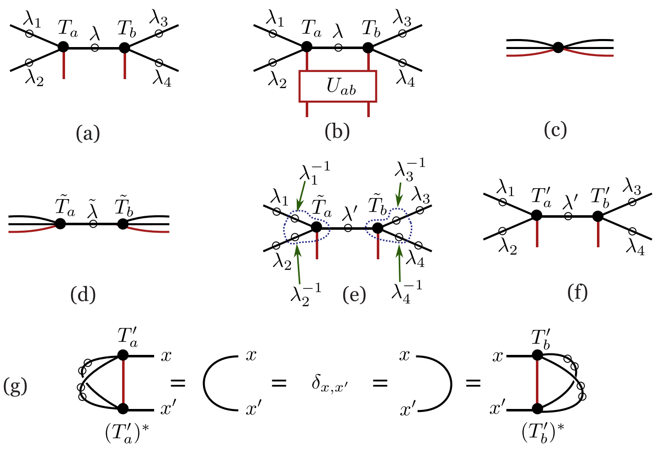


FIG. 9. The simple-update steps of applying a "gate" U_{ab} on the tensors $\{T_a, \lambda, T_b\}$ and updating them to $\{T'_a, \lambda', T'_b\}$. (a) The original tensors. (b) Applying U_{ab}. (c) The T_a, T_b tensors and all surrounding λ weights are contracted into one big tensor, which is reshaped as a matrix. (d) An SVD is performed on the matrix. (e) and (f) Trivial $\lambda_i\lambda_i^{-1}$ pairs are inserted into the external legs and define the new T'_a, λ', T'_b tensors. At this step, one can truncate the smallest weights of λ' to reduce the bond dimension back to D. (g) If no truncation was done, the resulting tensors satisfy some of the local canonical conditions.

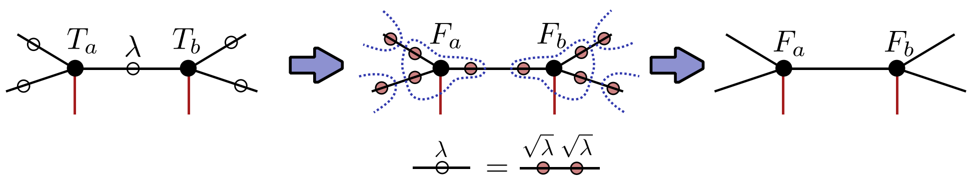


FIG. 10. Swallowing the λ weights in the T tensors and obtaining an equivalent TN with F tensors. The empty circles denote a simple-update weight tensor $\lambda_x\delta_{xy}$, and the red circles denote its square root: $\sqrt{\lambda_x}\delta_{xy}$.

When the operator U_{ab} is not unitary (e.g., in the case of imaginary time evolution), or when truncation is performed, the resultant TN will no longer be quasicanonical. Some of the canonical conditions will be satisfied, but not all of them. However, also in that case, the λ weights still provide a reasonable approximation for the local environments, as is evident by the success of the SU algorithm in many cases. Moreover, in the imaginary time case, if the system reaches a fixed point (the approximate ground state), it is also a fixed point of all the local SU steps. This state satisfies *all* the local canonical conditions and is therefore a quasicanonical state.

Finally, in the trivial-SU algorithm, we only apply the SVD steps, without applying U_{ab} [step (b) in Fig. 9] or performing the truncation. In other words, we essentially perform an SVD of the fused tensor in Fig. 9(d), which yields tensor T'_a, λ', T'_b satisfying the local canonical condition of Fig. 9(g). Repeating this step on all edges, if a fixed point is reached, it is by definition a quasicanonical PEPS and can be used to calculate local RDMs via the λ weights [see Fig. 3(c)]. Importantly, for the proof of Lemma 1, in this process the underlying PEPS keeps changing while still representing exactly the same quantum state $|\psi\rangle$.

We conclude this Appendix by noting that the trivial-SU algorithm has already been used previously under different names. See, for example, the quasiorthogonalization in Appendix B of Ref. [33] or the superorthogonalization in Ref. [34]. See also interesting parallels between our BP construction, the trivial-SU algorithm, and the network contractor dynamics (NCD) theory of Ref. [51] (see also Chap. 5 in Ref. [49]).

APPENDIX C: PROOFS OF LEMMAS 1 AND 2

1. Proof of Lemma 1

Assume that a trivial-SU step changes the TN $\mathcal{T}$ to $\mathcal{T}'$ by locally changing the adjacent tensors T_a, λ, T_b to T'_a, λ', T'_b, while keeping the rest of the tensors fixed [see Figs. 9(a)–9(f) with trivial $U_{ab} = \mathbb{1}$]. To simplify the bookkeeping, we "swallow" the λ tensors in the T_a tensors by splitting every λ tensor into $\lambda = \sqrt{\lambda} \cdot \sqrt{\lambda}$ and contracting each $\sqrt{\lambda}$ with its adjacent T_a tensor; see Fig. 10. We denote the resulting tensor networks by $\mathcal{F}, \mathcal{F}'$ and note that their local tensors are identical except for F_a, F_b and F'_a, F'_b, which are equal to the T_a, T_b, T'_a, T'_b tensors contracted with the appropriate $\sqrt{\lambda}$ tensors. The fact that the contraction of (T_a, λ, T_b) is equal to the contraction of (T'_a, λ', T'_b) implies that the contraction of (F_a, F_b) is equal to the contraction of (F'_a, F'_b).

Let $\{m_{a\to b}(x, x')\}$ be fixed-point BP messages of $\mathcal{F}$. We will use these messages to construct fixed-point BP messages $\{m'_{a\to b}(x, x')\}$ of $\mathcal{F}'$ that give the same RDMs. All messages except for the $a \to b$ and $b \to a$ messages remain the same. The $a \to b$ and $b \to a$ messages are defined by the BP iterative equations using the new tensors F'_a, F'_b so that they will satisfy them. For example, if tensor F_a is connected also to tensors F_c, F_d in addition to F_b, then $m'_{a\to b}(x, x')$ is given by the diagram in Fig. 1(c), replacing T tensors by corresponding F' tensors. To finish the proof, we need to show that this new set of messages (i) is a BP fixed point and (ii) produces the same RDMs according to the BP formula (see Fig. 2). Clearly, for adjacent vertices that have nothing to do with a, b, both conditions hold trivially, as the relevant messages

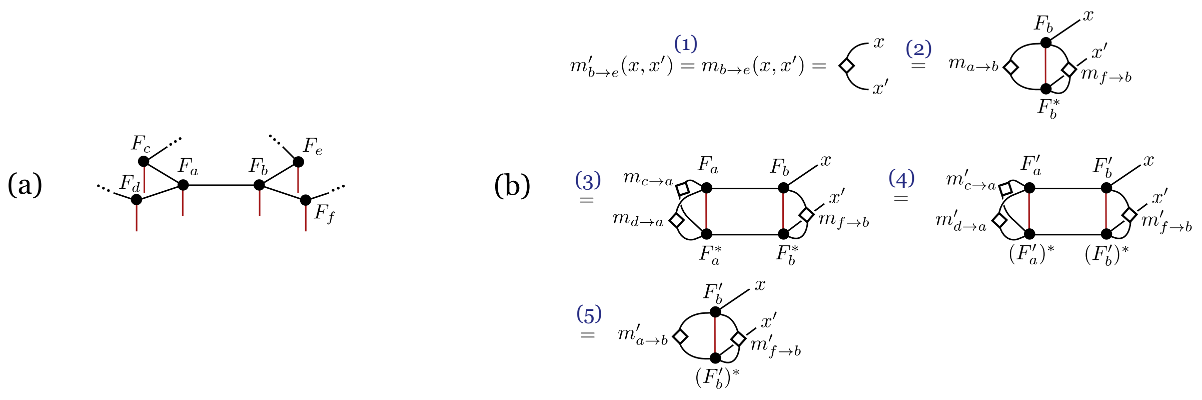


FIG. 11. Verifying point (i) in the proof of Lemma 1. Consider a patch in the TN given in (a), where a single trivial SU step was performed on the a, b vertices, changing T_a, λ, T_b to T'_a, λ', T'_b, or equivalently F_a, F_b to F'_a, F'_b. In (b) we verify that the new BP message $m'_{b\to e}(x, x')$ is related to the messages $m'_{a\to b}(x, x')$, $m'_{f\to b}(x, x')$ using the BP equations: Equality (1) follows from definition, $m'_{b\to e} = m_{b\to e}$. Then in (2) we use the assumption that $m_{b\to e}$ is a fixed point of the BP equation, and similarly in (3) we use that assumption on $m_{a\to b}$. In (4) we use the fact that the contraction of F_a, F_b is equal to the contraction of F'_a, F'_b, together with the definitions that all the new messages are equal to the old messages, except for the $a \leftrightarrow b$ messages. Finally, in (5) we use the definition of $m'_{a\to b}$, which satisfies the BP equations.

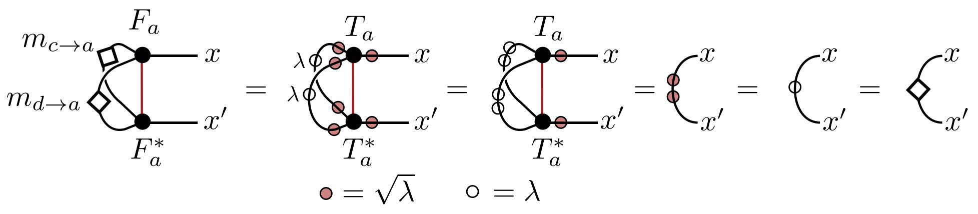


FIG. 12. The proof of Lemma 2: Defining the BP messages with Eq. (C1) and using the canonical condition [see Fig. 9(g)] show that these messages are fixed points of the BP equations.

and underlying tensors are unchanged. Let us then verify these points for vertices in the vicinity of a, b.

(a) Checking point (i). By definition, the $a \to b$ and $b \to a$ messages satisfy the BP equations. So we only need to verify that other messages from a or b (but not between them) satisfy the BP equations. Consider, for example, the message $b \to e$ in Fig. 11(a). We need to verify that $m'_{b\to e}(x, x')$ is indeed a BP fixed point, given as the appropriate expression of $m'_{a\to b}, m'_{f\to b}$ [see Fig. 1(c) for the BP equations]. This is proved in Fig. 11(b) in a series of five simple equalities (see the caption for full explanation), which rely on the facts that the original messages are fixed points of the BP equations and that the contraction of F_a, F_b is equal to the contraction of F'_a, F'_b.

(b) Checking point (ii). By definition, if we are interested in two-body RDMs on vertices that are different from both a and b, then the RDM estimate will remain the same because neither the relevant messages nor the tensors changed. We only need to verify this claim for the RDM ρ_{ab} and RDMs that contain a or b with another adjacent node, such as ρ_{be}. For the former, $\rho_{ab} = \rho'_{ab}$ because it depends on the incoming messages to the a, b nodes (which remain the same), together with the contraction of F'_a, F'_b, which by assumption is identical to that of F_a, F_b. For the latter, the proof uses the same idea as in checking point (i). Using the assumptions that the contraction of F_a, F_b is identical to that of $F'_a F'_b$ and that $F'_e = F_e$, it is easy to show that the three-body RDM ρ_{abe} is identical to that of ρ'_{abe}, from which we deduce that $\rho_{be} = \rho'_{be}$. This concludes the proof of Lemma 1.

2. Proof of Lemma 2

As in the first lemma, we first define $\mathcal{F}$ to be an equivalent TN in which every λ weight tensor in $\mathcal{T}$ was split into $\sqrt{\lambda} \cdot \sqrt{\lambda}$ and the $\sqrt{\lambda}$ tensors are contracted into the T_a tensors to give the F_a tensors (see Fig. 10). Next we define a set of messages

$$m_{a\to b}(x, x') \overset{\text{def}}{=} \lambda_x \delta_{x,x'} \qquad \text{(C1)}$$

for every two adjacent vertices a, b, where λ is the weight on the ab edge in the original $\mathcal{T}$ tensor. We claim that (i) these messages are BP fixed points on the $\mathcal{F}$ TN and (ii) they give the same two-body RDMs as those of the trivial-SU method of quasicanonical $\mathcal{T}$. Both claims are immediate. Claim (i) follows by writing the BP equation for the $a \to b$ message in terms of the F_a tensor and noticing that this expression is equal to $\lambda_x \delta_{x,x'}$ using the canonical condition on $\mathcal{T}$. This is illustrated in Fig. 12. Claim (ii) follows from definitions of the two-body RDMs of the BP method and the SU method [see Figs. 2 and 3(c)].

APPENDIX D: NUMERICAL TESTS

Here, we give the details of the numerical tests we performed to compare the convergence speed of the BP and trivial-SU algorithms on PEPS tensor networks. We tested the algorithms over two types of systems: (i) random PEPS and (ii) PEPS ground states of the antiferromagnetic Heisenberg model (AFH) with random couplings

$$H = \sum_{\langle a,b\rangle} J_{ab} \boldsymbol{\sigma}_a \otimes \boldsymbol{\sigma}_b, \quad J_{ab} < 0. \qquad \text{(D1)}$$

Both systems were simulated on 4×4 and 10×10 square lattices. In all tests, the physical bond dimension was $d = 2$, and virtual bond dimensions were $D = 2, 3, 4$. All in all, we therefore tested $2 \times 2 \times 3 = 12$ different configurations. For every configuration we used statistics of 20–50 different random realizations on which we did the analysis.

In the random PEPS configurations, the tensor entries where chosen as $a + ib$, where a, b were uniformly distributed in $(-1, 1)$. In the AFH configurations, we used random couplings J_{ab} uniformly distributed in the interval $(-1, 0)$. To obtain the ground states of these models, we ran an imaginary time evolution with the simple-update algorithm, decreasing values of imaginary time steps by $\delta\tau = 0.1, \dots, 0.0001$. After obtaining an approximation to the ground state, we applied a random local gauge change on every bond in order to get a TN that is far away from a canonical form. Specifically, for every virtual edge (a, b), we drew a random matrix V_{ab} which was a product of a random unitary with a random diagonal matrix with entries between 0.5 and 2. We then inserted the identity $V_{ab}^{-1} V_{ab} = \mathbb{1}$ in the middle of the edge, absorbing V_{ab}^{-1}

TABLE I. Average ratio of convergence times $T_{\text{BP}}/T_{\text{tSU}}$, together with standard deviations, for random PEPS and ground states of the antiferromagnetic Heisenberg model (AFH) with random nearest-neighbor couplings on $N \times N$ lattices. We simulated different bond dimensions $D = 2, 3, 4$ using 20–50 realizations for each configuration. Full details can be found in Appendix D.

	Random PEPS		AFH	
	4×4	10×10	4×4	10×10
$D = 2$	0.7 ± 0.2	0.6 ± 0.2	1.4 ± 0.1	1.3 ± 0.1
$D = 3$	0.7 ± 0.2	0.7 ± 0.2	1.1 ± 0.2	1.2 ± 0.1
$D = 4$	0.7 ± 0.2	0.8 ± 0.2	1.6 ± 0.5	1.8 ± 0.2

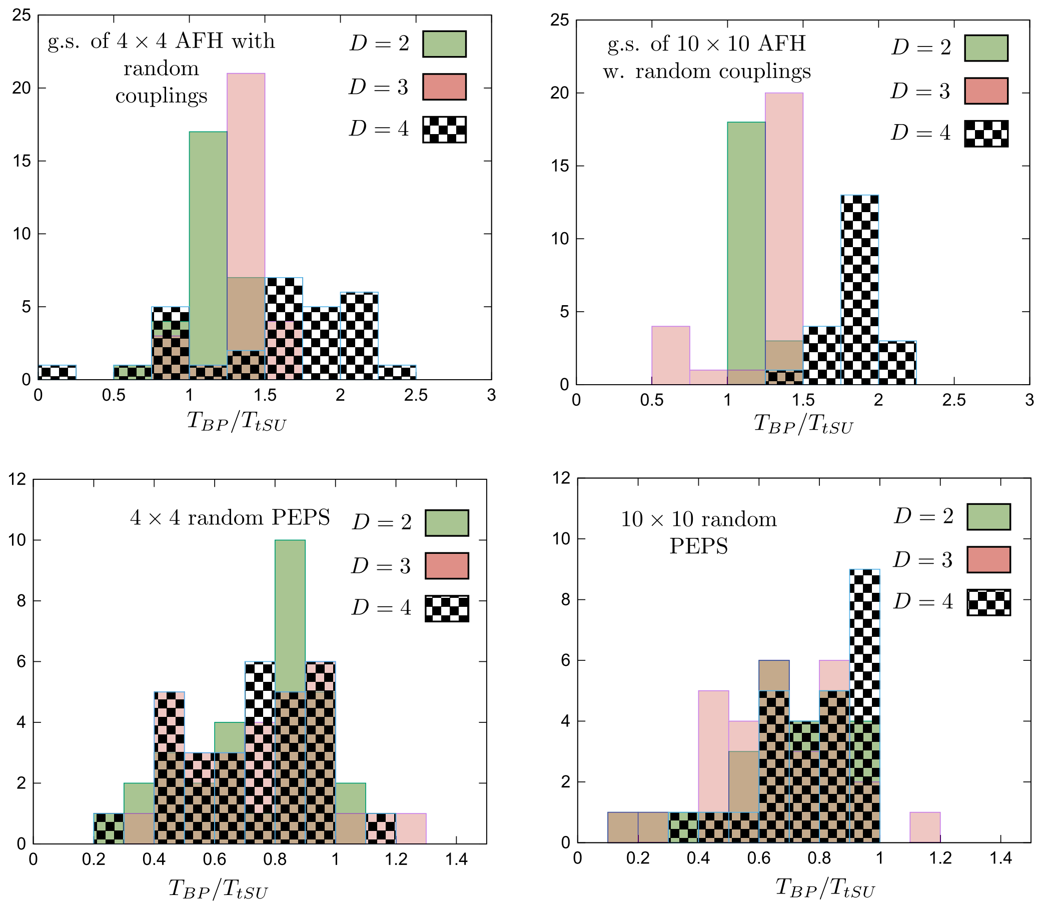


FIG. 13. The ratio of the number of iterations that it takes for the BP algorithm to converge to that for the trivial-SU algorithm. The tests were on four different systems: 4×4 and 10×10 antiferromagnetic Heisenberg models with random coupling, as well as 4×4 and 10×10 random PEPS. For each of these four cases, PEPS were used with bond dimension $D = 2, 3, 4$, and the statistics was generated using 20–50 different realizations. More details on the numerical procedure can be found in the main text. g.s., ground state; w., with.

in T_a and V_{ab} in T_b. This way, the resultant TN was far from quasicanonical, yet represented the same approximate ground state.

To calculate the convergence time for each algorithm, we looked at the averaged trace distance of two-body RDMs between consecutive iterations [see Fig. 3(c) for a trivial-SU RDM illustration and Fig. 2 for a BP RDM illustration]. We stopped the iterations when $\frac{1}{m}\sum_{\langle a,b\rangle}\|\rho_{ab}^{(t+1)} - \rho_{ab}^{(t)}\|_1 < 10^{-6}$, where $\langle a, b\rangle$ denotes nearest-neighbor nodes and m is the total number of such neighbors. We defined T_{BP} and T_{tSU} to be the convergence times of the two algorithms and calculated the T_{BP}-to-T_{tSU} ratio statistics, which are given in Table I and Fig. 13. Evidently, the BP performs slightly better for random PEPS, where there are presumably very few correlations between the different spins. On the other hand, for the more ordered instances of the AFH ground states, the trivial-SU algorithm seems to converge faster, and this trend seems to increase as we increase the bond dimension D.

[1] R. Orús, Ann. Phys. (Amsterdam) **349**, 117 (2014).
[2] M. J. Wainwright and M. I. Jordan, Found. Trends Mach. Learn. **1**, 1 (2008).
[3] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques* (MIT Press, Cambridge, MA, 2009).
[4] M. Mezard and A. Montanari, *Information, Physics, and Computation* (Oxford University Press, Oxford, 2009) .
[5] G. F. Cooper, Artif. Intell. **42**, 393 (1990).
[6] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac, Phys. Rev. Lett. **98**, 140506 (2007).
[7] I. L. Markov and Y. Shi, SIAM J. Comput. **38**, 963 (2008).
[8] R. Orús and G. Vidal, Phys. Rev. B **80**, 094403 (2009).
[9] T. Nishino and K. Okunishi, J. Phys. Soc. Jpn. **65**, 891 (1996).
[10] R. J. Baxter, J. Math. Phys. (Melville, NY) **9**, 650 (1968).
[11] R. J. Baxter, J. Stat. Phys. **19**, 461 (1978).
[12] A. W. Sandvik and G. Vidal, Phys. Rev. Lett. **99**, 220602 (2007).
[13] L. Wang, I. Pižorn, and F. Verstraete, Phys. Rev. B **83**, 134421 (2011).
[14] A. J. Ferris and G. Vidal, Phys. Rev. B **85**, 165146 (2012).
[15] M. Levin and C. P. Nave, Phys. Rev. Lett. **99**, 120601 (2007).

[16] E. Efrati, Z. Wang, A. Kolan, and L. P. Kadanoff, Rev. Mod. Phys. **86**, 647 (2014).
[17] E. Robeva and A. Seigal, Inf. Inference **8**, 273 (2018).
[18] H. C. Jiang, Z. Y. Weng, and T. Xiang, Phys. Rev. Lett. **101**, 090603 (2008).
[19] S. S. Jahromi and R. Orús, Phys. Rev. B **99**, 195105 (2019).
[20] J. Pearl, in *Proceedings of the National Conference on Artificial Intelligence* (AAAI, Menlo Park, CA, 1982), pp. 133–136.
[21] F. R. Kschischang, B. J. Frey, and H. Loeliger, IEEE Trans. Inf. Theory **47**, 498 (2001).
[22] H. A. Bethe, Proc. R. Soc. London, Ser. A **150**, 552 (1935).
[23] R. Peierls, Proc. Cambridge Philos. Soc. **32**, 477 (1936).
[24] J. C. Bridgeman and C. T. Chubb, J. Phys. A: Math. Theor. **50**, 223001 (2017).
[25] See, for example, Theorem 14.1 in Ref. [4].
[26] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, IEEE J. Sel. Areas Commun. **16**, 140 (1998).
[27] J. S. Yedidia, W. T. Freeman, and Y. Weiss, in *Advances in Neural Information Processing Systems* (Kaufmann, San Mateo, CA, 2001), Vol. 13, pp. 689–695.
[28] M. X. Cao and P. O. Vontobel, in *2017 IEEE Information Theory Workshop (ITW)* (IEEE, Piscataway, NJ, 2017), pp. 136–140.
[29] R. Alkabetz, Using the belief propagation algorithm for finding tensor networks approximations of many-body ground states, Master's thesis, Technion, Haifa, Israel, 2020.
[30] G. Vidal, Phys. Rev. Lett. **91**, 147902 (2003).
[31] G. Vidal, Phys. Rev. Lett. **93**, 040502 (2004).
[32] A. J. Daley, C. Kollath, U. Schollwöck, and G. Vidal, J. Stat. Mech.: Theory Exp. (2004) P04005.
[33] H. Kalis, D. Klagges, R. Orús, and K. P. Schmidt, Phys. Rev. A **86**, 022317 (2012).
[34] S.-J. Ran, W. Li, B. Xi, Z. Zhang, and G. Su, Phys. Rev. B **86**, 134429 (2012).
[35] A. Al-Bashabsheh and Y. Mao, IEEE Trans. Inf. Theory **57**, 752 (2011).
[36] R. Kikuchi, Phys. Rev. **81**, 988 (1951).
[37] T. Morita, M. Suzuki, K. Wada, and M. Kaburagi, Prog. Theor. Phys. Suppl. **115**, 136 (1994).
[38] A. Montanari and T. Rizzo, J. Stat. Mech.: Theory Exp. (2005) P10011.
[39] M. Chertkov and V. Y. Chernyak, J. Stat. Mech.: Theory Exp. (2006) P06009.
[40] J. Mooij, B. Wemmenhove, B. Kappen, and T. Rizzo, in *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research Vol. 2 (PMLR, Cambridge, MA, 2007), pp. 331–338.
[41] E. Nachmani, Y. Be'ery, and D. Burshtein, in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)* (IEEE, Piscataway, NJ, 2016), pp. 341–346.
[42] G. T. Cantwell and M. E. J. Newman, Proc. Natl. Acad. Sci. USA **116**, 23398 (2019).
[43] Z.-C. Gu, M. Levin, and X.-G. Wen, Phys. Rev. B **78**, 205116 (2008).
[44] J. Jordan, R. Orús, G. Vidal, F. Verstraete, and J. I. Cirac, Phys. Rev. Lett. **101**, 250602 (2008).
[45] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Kaufmann, San Mateo, CA, 1988).
[46] A. Critch and J. Morton, Symmetry Integrability Geom. Methods Appl. **10**, 095 (2014) .
[47] J. Chen, S. Cheng, H. Xie, L. Wang, and T. Xiang, Phys. Rev. B **97**, 085104 (2018).
[48] H. Loeliger and P. O. Vontobel, IEEE Trans. Inf. Theory **63**, 5642 (2017).
[49] S.-J. Ran, E. Tirrito, C. Peng, X. Chen, L. Tagliacozzo, G. Su, and M. Lewenstein, *Tensor Network Contractions: Methods and Applications to Quantum Many-Body Systems* (Springer Nature, Cham, 2020) .
[50] Specifically, some of the conditions are exactly satisfied when *no* truncation is done; otherwise, there is an error which scales like the sum of the squares of the Schmidt coefficients that were neglected.
[51] S.-J. Ran, B. Xi, T. Liu, and G. Su, Phys. Rev. B **88**, 064407 (2013).