Block belief propagation algorithm for two-dimensional tensor networks

Chu Guo,[1] Dario Poletti,[2,3,4,5] and Itai Arad[6,*]

[1]*Key Laboratory of Low-Dimensional Quantum Structures and Quantum Control of Ministry of Education, Department of Physics and Synergetic Innovation Center for Quantum Effects and Applications, Hunan Normal University, Changsha 410081, China*
[2]*Science, Mathematics and Technology Cluster, Singapore University of Technology and Design, 8 Somapah Road, 487372, Singapore*
[3]*Engineering Product Development Pillar, Singapore University of Technology and Design, 8 Somapah Road, 487372, Singapore*
[4]*Centre for Quantum Technologies, National University of Singapore, 117543, Singapore*
[5]*MajuLab, CNRS-UNS-NUS-NTU International Joint Research Unit, UMI 3654, Singapore*
[6]*Faculty of Physics, Technion, Haifa 3200003, Israel*

(Received 5 March 2023; revised 16 August 2023; accepted 23 August 2023; published 6 September 2023)

Belief propagation is a well-studied algorithm for approximating local marginals of multivariate probability distribution over complex networks, while tensor network states are powerful tools for quantum and classical many-body problems. Building on a recent connection between the belief propagation algorithm and the problem of tensor network contraction, we propose a block belief propagation algorithm for contracting two-dimensional (2D) tensor networks and approximating the ground state of 2D systems. The advantages of our method are threefold: (1) the same algorithm works for both finite and infinite systems; (2) it allows natural and efficient parallelization; and (3) given its flexibility, it would allow us to deal with different unit cells. As applications, we use our algorithm to study the 2D Heisenberg and transverse Ising models, and show that the accuracy of the method is on par with state-of-the-art results.

DOI: 10.1103/PhysRevB.108.125111

I. INTRODUCTION

Two-dimensional quantum systems represent an important class of problems of long-lasting theoretical and practical interest. Many such systems are notoriously difficult to investigate numerically, especially in the strongly correlated regime. For this reason, developing efficient and accurate numerical methods is always in need. Quantum Monte Carlo [1] and tensor network (TN) methods [2] have both been demonstrated to be successful approaches for 2D systems in the past decades. However, the quantum Monte Carlo methods often suffer from the sign problem [3]. Projected entangled pair states (PEPSs) are a class of two-dimensional TNs [2,4–6] that can be used to study both finite (fPEPS) and infinite (iPEPS) 2D systems. In particular, the iPEPS algorithms have attracted great attention during the last 20 years because of their lack of a sign problem and their ability to deal with strongly correlated systems [7–13]. A key challenge for PEPS algorithms is an efficient and stable procedure for computing expectation values of local observables, which can become a #P-hard problem in the worst cases in 2D and higher dimensions [14]. In a number of physical scenarios, the problem tends to be more tractable but it still requires high computational resources and advanced numerical schemes. Existing approaches include, for example, the boundary matrix product state (bMPS) method [4,15], the corner transfer matrix (CTM) method [16,17], Monte Carlo sampling [18,19], and variants of TN renormalization schemes [20–22].

Borrowing ideas from classical probabilistic models, we propose an alternative method for (approximately) contracting two-dimensional TNs, which is highly parallelizable and is flexible to deal with both finite and infinite 2D systems. There is a close similarity between quantum and classical many-body systems and multivariate probabilistic models. In both cases, the states live in spaces of exponential dimension and it is important to (approximately) compute local properties of the system: local expectation values in the former cases and local marginals in the latter case [23]. Therefore, methods developed in one field could often benefit the other. Belief propagation (BP) is a well-established statistical inference algorithm for computing local marginals of multivariate probabilistic models on complex networks, which is known to be exact for tree networks [24]. BP has been widely applied to very diverse areas, ranging from Bayesian inference [24], statistics physics [25–27], combinatorial optimizations [28,29], and epidemic spreading [30] to name a few. For networks that contain a large number of loops or long-range correlations, BP could result in poor performance since it only considers the direct neighbors of each node and treats the environment of a node in a separable way. This shortcoming has been the subject of extensive research that resulted in a plethora of algorithms that generalize and improve BP in different ways [31–41]. Broadly speaking, these algorithms clump together neighboring nodes and treat small loops exactly at the expense of higher computational resources and elaborated bookkeeping.

Attempts to use BP to approximate TN contraction were presented in Refs. [42,43]. When applied with the imaginary time evolution (ITE) to PEPS, it was shown to be equivalent to the simple-update algorithm [42], in which the environment is approximated in a separable, mean-field way [21]. In this paper, we present a generalization of the BP algorithm of Ref. [42], which we call *block belief propagation* (blockBP).

*arad.itai@fastmail.com

The algorithm partitions the system into nonoverlapping blocks of spins, and uses the framework of Ref. [42] to define BP messages between these blocks. From the converged messages, one can obtain an approximation of the local environment of every block, which is far superior to the mean-field environment of the original BP, since the loops within each block are accounted for. We benchmark our algorithm on the 2D Heisenberg model and transverse Ising (TI) model, where we show that it can reach similar precision to current state-of-the-art methods. Importantly, we show that our approach can be easily parallelized, thus leading to a much higher computational efficiency compared to other TN-based state-of-the-art methods. Furthermore, we also demonstrate that the same algorithm can be readily used both for finite systems and infinite systems with translational invariance. Lastly, the algorithm can be applied to different unit cells and systems with different geometries. As such, blockBP might also be potentially used in quantum chemistry problems with less regular structures [44].

II. THE blockBP ALGORITHM

To present the blockBP algorithm, we assume basic familiarity with TN such as MPS and PEPS [2,45]. More information for the unfamiliar reader can be found in the Appendixes A and B. Consider a 2D many body whose state is given by a PEPS $|\psi\rangle = \sum_{i_1,\ldots,i_n} \mathrm{Tr}(T_1^{i_1}\cdots T_n^{i_n})|i_1,\ldots,i_n\rangle$. Here, $T_1,\ldots,T_n$ represent the local tensors of the PEPS with dimension of the virtual legs, also known as bond dimension, D, and $i_1,\ldots,i_n$ denote the physical legs. Virtual legs are omitted for brevity and $\mathrm{Tr}(\cdots)$ denotes TN contraction.

The blockBP algorithm is an efficient method to approximate the local environments in a 2D PEPS. The first step of the algorithm is to move to a double-layer PEPS. This is a closed TN that represents the scalar $\langle\psi|\psi\rangle$ by contracting the physical legs between a bra TN $\langle\psi|$ and a ket TN $|\psi\rangle$—see Figs. 1(a) and 1(b). Next, we partition the system into nonoverlapping blocks as in Fig. 1(c). The goal of the blockBP algorithm is to simultaneously find an approximation to the environment of each block. The exact environment can be written as a periodic MPS that surrounds the block [Fig. 1(e)], which is equal to the contraction of the entire doubled-layered PEPS *outside* the block. Knowing the environment of each block allows an easy calculation of the expectation values within each block, as well as an optimal update of the tensors of the block during an ITE or variational algorithms. However, calculating it directly might be a daunting task when the system size is large.

The key approximation of the blockBP algorithm is to break the block environment into a product of MPSs, where each MPS belongs to a block edge corresponding to an adjacent block [see Figs. 1(e) and 1(f)]. This inflicts an unavoidable error on the environment, as it breaks some of its entanglement, but in many cases this error is negligible (see discussion on this point below).

The MPS states that make each environment are calculated in a self-consistent way using a BP algorithm. We define a message-passing algorithm in which at step ℓ each block receives (sends) MPS messages from (to) its adjacent blocks along their common edge. We denote by $m^{(\ell)}_{A\to B}$ the MPS

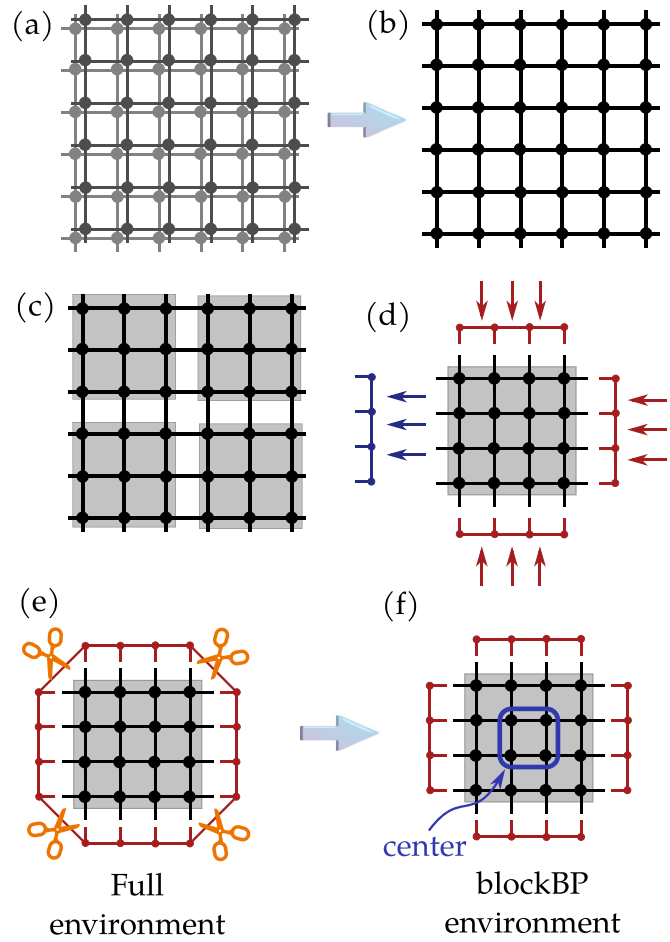

FIG. 1. (a), (b) Forming a double layer 2D PEPS by contracting the physical legs of $|\psi\rangle$ and $\langle\psi|$. (c) Partitioning of the system into blocks (grey regions). (d) Computing the left output MPS message for a given block. (e) The exact block environment. (f) The blockBP environment, which approximates the exact block environment using a product of the converged incoming MPS messages. These can be used to compute the local environments inside the center of the block.

message from block A to block B at the ℓth iteration. Let T_A denote the part of the double layer PEPS that corresponds to block A. This is an open TN with double-layer legs at each edge of the block. The MPS message from block A to block B at the $\ell+1$ iteration is obtained by contracting T_A with all incoming MPS messages to block A of the ℓth iteration, except for the message from B [see Fig. 1(d)]:

$$m^{(\ell+1)}_{A\to B} = \mathrm{Tr}\left(T_A \prod_{A'\in N_A\setminus\{B\}} m^{(\ell)}_{A'\to A}\right). \tag{1}$$

When the underlying blocks graph is a tree, then just as in ordinary BP, the fixed point of Eq. (1) describes the exact environment of each block. Note that, indeed, in such a case, the environment of each block breaks into a product of MPSs.

The iteration in Eq. (1) can be done efficiently using, e.g., the bMPS method. In addition, messages of different blocks can be computed in parallel. We stop the iterations when the average distance between the normalized MPS messages at

two consecutive iterations is sufficiently small (say, smaller then $\epsilon = 10^{-5}$). This usually happens in less than ten steps.

For blocks of size 1, our algorithm reduces to the BP contraction algorithm of Ref. [42], which has been shown to be equivalent to the simple-update algorithm. These algorithms are exact on treelike (i.e., loop free) TNs, but may fail on more complicated graphs. In contrast, for larger block sizes blockBP can yield very accurate correlations inside a block, as it considers exactly the loops within it. We note, however, that as the entanglement around every block is broken between every two adjacent block edges, some inaccuracies are unavoidable. In noncritical systems, this problem can be largely mitigated if we consider only observables in the *center* of a block [see Fig. 1(e)]. For these sites, most of the correlations in the local environment come from nearby tensors, which lie inside the block and are therefore properly accounted for. By increasing the block size, the accuracy for the center sites can be improved at the expense of higher computational costs. In this respect, our algorithm shares some similarities with the cluster-update algorithm [46], where the local environment of row i is calculated by contracting a mean-field environment of the rows at distance δ from i, together with the full TN of the rows $i-\delta, \ldots, i+\delta$ that surround i, although we do not need to rely on a mean-field environment.

The relation between the block size, the system size, and the center size can be chosen differently depending on the specific tasks to balance the computational efficiency and the accuracy. In addition, since we only calculate the local environment of spins at the center of each block, we need several partitions of the system in blocks to cover all the bonds (neighboring spins) in the system. Bookkeeping the different partitions becomes much easier if we work with periodic boundary conditions (PBCs), in which different partitions correspond to shifting the lattice in different directions. This technique is useful also in the case of open boundary conditions (OBCs), as we can embed an OBC PEPS in a PBC PEPS by setting the bond dimension at the boundaries to 1. Furthermore, we can easily adjust it for an infinite, translational invariant PEPS described by tensors in a unit cell (iPEPS). In such case, one block is sufficient to calculate the environment of a unit cell at its center. The block sends and receives messages to itself, which mimics the case where it is surrounded by its own copies.

To efficiently perform the contractions in Eq. (1), we use the bMPS method with a bond dimension χ_m, which we typically set to $\chi_m = D^2$. Once the messages have converged, we use them to calculate the local environment of spins in the center of a block so as to compute observables or optimize the tensors in the center to approach the ground state. To this aim, we use the bMPS algorithm once more, this time with a larger bond dimension χ, which we set to $\chi = 2D^2 + 10$ if not specified otherwise.

In this paper, to obtain the ground state, we use blockBP in combination with ITE, and we refer to this as blockBP update. More details can be found in Appendix B. Note that in our algorithm we make use of bMPS, but only over small portions of the system, our blocks, and not on the entirety of the system as in the bMPS algorithm.

Before presenting our numerical results, we summarize the important features of blockBP: (1) It works for both finite and infinite 2D systems, as Eq. (1) is agnostic about the boundaries, and the differences between them are automatically revealed when computing messages. For finite systems, whenever the boundary is met, the information before the boundary will be lost because the virtual dimension of the tensor at the boundary is 1. Without a boundary, instead the fixed point will directly correspond to the infinite limit. (2) The overall complexity of blockBP is that of bMPS times a constant factor due to the overhead of computing the messages. However, Eq. (1) can be evaluated independently for each block, and after the messages have converged, the calculation of local environments inside the center is also completely independent. Since these are the most computation-intensive parts of the algorithm, blockBP can be efficiently parallelized on distributed architectures. (3) blockBP is flexible on the choice of the unit cells (size and geometry), which is particularly useful for infinite systems with large unit cells. In the following, we demonstrate the efficiency and accuracy of blockBP for both finite and infinite 2D systems.

TABLE I. Ground-state energies of the 21×21 transverse Ising model with $B = 2.5, 3, 3.5$. We used a block size of 7×7 (nine blocks in total) for blockBP.

	D	2.5	3	3.5
bMPS [47]	2	−2.77340(2)	−3.18128(6)	−3.64849(3)
	3	−2.77346(1)	−3.18242(1)	−3.64873(1)
	4	−2.77346(1)	−3.18243(1)	−3.64873(1)
blockBP	2	−2.77334(7)	−3.18121(9)	−3.64842(7)
	3	−2.77346(3)	−3.18242(3)	−3.64872(9)
	4	−2.77346(4)	−3.18244(4)	−3.64873(1)

III. BLOCKBP FOR FINITE SYSTEMS

We first demonstrate the accuracy of blockBP by applying it to compute the ground state of the finite-size TI model, whose Hamiltonian is $H_{\mathrm{TI}} \stackrel{\mathrm{def}}{=} -\sum_{\langle i,j\rangle} \sigma_i^z \sigma_j^z - B \sum_k \sigma_k^x$ with B the strength of the transverse field, and antiferromagnetic Heisenberg (AFMH) model with Hamiltonian $H_{\mathrm{AFH}} \stackrel{\mathrm{def}}{=} \sum_{\langle i,j\rangle} (\sigma_i^x \sigma_j^x + \sigma_i^y \sigma_j^y + \sigma_i^z \sigma_j^z)/4$. Both models are defined on a square lattice with nearest-neighbor interactions. For the ITE of finite systems, we chose the center size to be the same as the block size for efficient update. The final energies for the finite systems are computed using bMPS. The energies per site of the final PEPS are shown in Table I for TI and in Table II for AFH, which are compared to the bMPS full-update results from Ref. [47]. We see that the blockBP energies are very accurate for all cases. For $D = 6$, the blockBP energies are slightly larger, which is likely because a larger block size should be used to account for longer range correlations. We use at most $D = 6$ due to the unfavorable scaling of the algorithm with D, namely, $O(D^{12})$, essentially because we use bMPS for each block.

The accuracy of PEPS algorithms relies on the accuracy of the computed local environment of each bond. To study this, we used blockBP to compute the reduced density matrices (RDMs) in the ground state of the TI model at $B = 2.5, 3.0, 3.5$ and compared it to the RDMs that were

TABLE II. Ground-state energies of the AFH model of sizes 10×10 and 14×14, for which We used block sizes of 5×5 and 7×7, respectively.

	10×10		14×14	
D	bMPS [47]	blockBP	bMPS [47]	blockBP
2	−0.61310(2)	−0.61310(1)	−0.62631(1)	−0.62631(3)
3	−0.61999(1)	−0.62012(0)	−0.63246(1)	−0.63273(4)
4	−0.62637(2)	−0.62636(2)	−0.63832(3)	−0.63831(8)
5	−0.62739(1)	−0.62737(7)	−0.63901(1)	−0.63905(2)
6	−0.62774(1)	−0.62759(3)	−0.63930(1)	−0.63928(3)

calculated using bMPS on the *same* PEPS. We used a system of 21×21 spins with OBCs, where the ground state was approximated by PEPS with $D = 2, 3, 4$ and was calculated using the bMPS full-update algorithm. For blockBP, we used a block size of 7×7 with a center of 2×2. To compare the blockBP and bMPS RDMs, we calculated the trace distance $D(\rho, \sigma) \stackrel{\text{def}}{=} \frac{1}{2}\|\rho - \sigma\|_1$ of two-local RDMs of the horizontal bonds. For $B = 2.5, 3.5$, which are away from the critical value ($B_c \approx 3.044$ [48]), the RDMs computed by these two methods were very close to each other with average distances lower than 10^{-5} for all the Ds considered, while for $B = 3.0$ we got, on average, a trace distance of $\approx 10^{-3}$ (see Appendix C for details on this, and Appendix D for further discussion on convergence). Interestingly, although the accuracy of the RDMs computed using our blockBP near the critical point is lower by about two orders of magnitude than the accuracy away from criticality, the ground-state energy of the resultant PEPS using blockBP update is as good as those from the bMPS full-update (Table I).

IV. PARALLEL COMPUTATION

In Fig. 2, we show the scaling of the runtime per ITE step (iteration) for blockBP with block size 5×5 against the number of processes (n_{proc}), for the 40×40 AFH model. We can clearly see that the runtime of blockBP scales almost linearly as $1/n_{\text{proc}}$ as n_{proc} increases up to $n_{\text{proc}} = 32$. Instead, bMPS acts sequentially on the whole system, and for this reason no approach to parallelize it has, to the best of our knowledge, been found. For $n_{\text{proc}} = 1, 2$, blockBP is slower than bMPS due to the overhead of computing the messages, but beyond these values of n_{proc} we see a clear advantage of using blockBP.

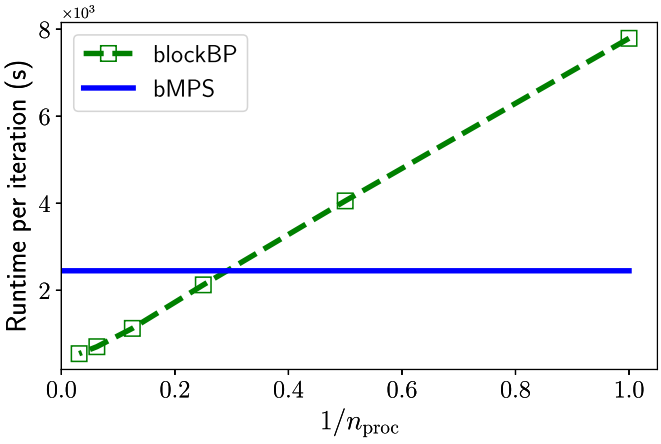

FIG. 2. Scaling of the blockBP runtime for an imaginary time evolution step versus the number of processes n_{proc} and for the AFH model of size 40×40, using a block size 5×5. The solid line shows the corresponding runtime of bMPS using a single thread for comparison.

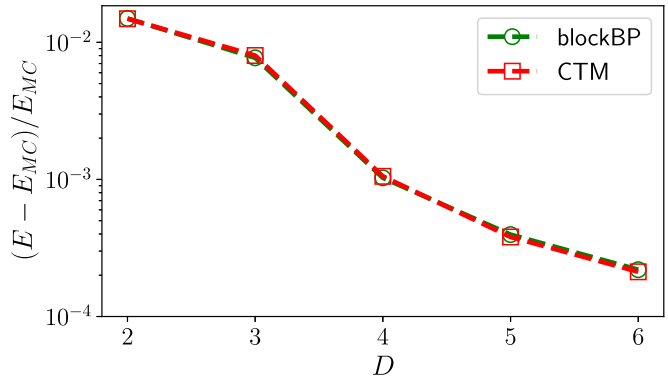

FIG. 3. Relative energy compared to the exact energy E_{MC} obtained by quantum Monte Carlo [49] for the infinite AFH model. The green dashed line with circle shows the results computed using BlockBP while the red dashed line with square shows the CTM results [50].

V. BLOCKBP FOR INFINITE SYSTEMS

Finally, we apply blockBP to study infinite quantum and classical systems. We first compute the ground state of the infinite AFH model as an iPEPS, for which we chose a unit cell size as 2×2 and a block size of 4×4. The final energies are computed by embedding the 2×2 iPEPS into a 52×52 finite PEPS with randomly initialized boundaries, and then computing the energy in the center using bMPS. We have verified the correctness of our energy estimate by checking that the resultant energies have well converged against the finite PEPS size. From the results in Fig. 3, we can see that our precision is on par with the CTM results from Ref. [50], while only requiring a computational complexity that is similar to the bMPS algorithm applied to just a 4×4 system.

In Fig. 4, we further study the infinite classical Ising model and the TI model to demonstrate the versatility of blockBP. In Fig. 4(a), we look at the local magnetization m_z of the Gibbs state of the 2D classical Ising model on a square lattice (with Hamiltonian $H = \sum_{\langle i,j \rangle} \sigma_i^z \sigma_j^z$) as a function of the inverse temperature β. This can be written exactly as a TN with $\chi = 2$, which is an ideal test ground for computing local observables since it does not require updating. In this case, we used infinite blockBP to calculate $m_z = \langle \sigma_z \rangle$ for the spin in the center of the block and consider the effect of different block sizes compared to Onsager's exact solution [51,52] (continuous line). We see that with block size 5×5, we already obtain very accurate results away from the critical point (with $\beta_c \approx 0.44$) and, increasing the block size, we obtain more accurate results near β_c. Here we note that our

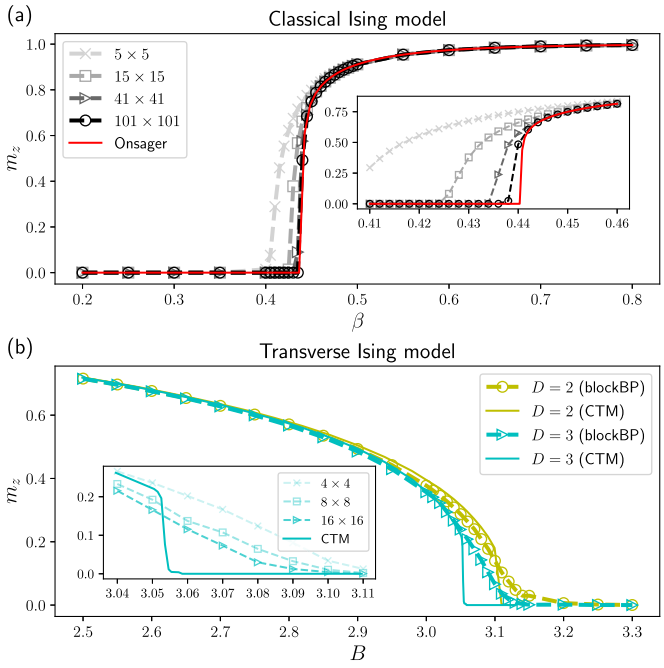

FIG. 4. (a) Local magnetization m_z as a function of the inverse temperature β for the infinite classical Ising model. The red solid line is Onsager's exact solution. (b) m_z as a function of B for the infinite quantum Ising model for $D = 2$ (orange dashed line with circle) and $D = 3$ (cyan dashed line with triangle), where we have used a 2×2 unit cell and a block size 4×4. The corresponding solid lines are results from Ref. [50]. The inset shows m_z computed using larger block sizes near the critical point for $D = 3$.

results close to β_c are not as accurate as those obtained in Refs. [13,20,22,53,54], however, they are more accurate than the results obtained using other variants of the BP algorithms such as in Refs. [38,39,55].

In Fig. 4(b), we used blockBP + ITE to compute the local magnetization $m_z = \langle \sigma_z \rangle$ at the ground state of the infinite TI model for different values of the external field B, and compared our results to those obtained by the CTM from Ref. [50]. We set the center size to be 2×2 and use a block size 4×4. To compute m_z for the final converged infinite system, we have copied the block into a 52×52 finite TN and computed the average m_z at the central 2×2 cell using bMPS. We can see that away from criticality ($B_c \approx 3.044$ [48]), our results agree well with the CTM results, especially for $D = 3$. For $D = 2$, our results approach better the results at $D = 3$ for $B < 3.1$. We note that with 4×4 block size, our simulation is extremely efficient ($\approx$0.4s and $\approx$1.1s per iteration for $D = 2$ and $D = 3$, respectively, using a single core of 2.3 GHz frequency). As in the classical Ising model, with a small block size we are not able to accurately reproduce m_z near B_c. Nevertheless, the results close to the phase transition can be systematically improved by increasing the block size.

VI. SUMMARY AND OUTLOOK

In summary, we have proposed a belief propagation algorithm for approximately contracting 2D TNs. Our approach is straightforwardly applicable for both finite and infinite

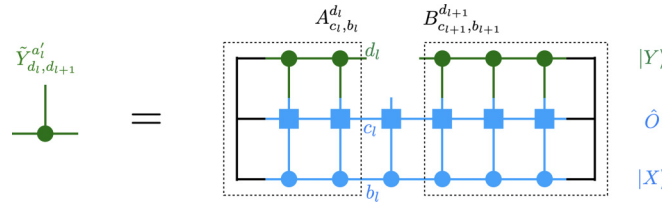

FIG. 5. Updating a local site tensor in the zip-up algorithm to multiply an MPS by an MPO, which corresponds to Eq. (A5). The left block is represented by the tensor $A^{d_l}_{c_l,b_l}$, while the right block by $B^{d_{l+1}}_{c_{l+1},b_{l+1}}$. The tensor $\tilde{Y}^{a'_l}_{d_l,d_{l+1}}$ is obtained contracting these two tensors with $O^{a'_l,a_l}_{c_l,c_{l+1}}$ and $X^{a_l}_{b_l,b_{l+1}}$.

systems, different unit cell sizes, and can be readily generalized to different geometries. Furthermore, our method allows straightforward and efficient parallelization. The accuracy and efficiency of the method are demonstrated with applications to prototypical quantum 2D systems, and benchmarked with state-of-the-art results based on ITE. Future directions of investigation include testing the algorithm on different lattices, including higher dimensions and/or different geometries, as well as using gradient-based methods to optimize the PEPS instead of ITE [56], as done for CTM [57,58].

ACKNOWLEDGMENTS

We are grateful to J. Hasik for his comments on a previous version of the paper and for providing us data of the iPEPS ground state (with $D = 3$) for the critical TI model obtained from gradient-based optimization, based on which we verified that with blockBP we need a large block size (100×100) as well as a larger χ_m ($\chi_m = 25$) to reach similar precision. C.G. acknowledges support from the Open Research Fund from the State Key Laboratory of High Performance Computing of China (Grant No. 202201-00). I.A. acknowledges the support of the Israel Science Foundation (ISF) under the Individual Research Grant No. 1778/17 and joint Israel-Singapore NRF-ISF Research Grant No. 3528/20. D.P. acknowledges support from joint Israel-Singapore NRF-ISF Research Grant No. NRF2020-NRF-ISF004-3528.

APPENDIX A: DETAILS OF THE BLOCKBP IMPLEMENTATION

The two building blocks of our blockBP algorithm are (1) computing the fixed points of the MPS messages and (2)

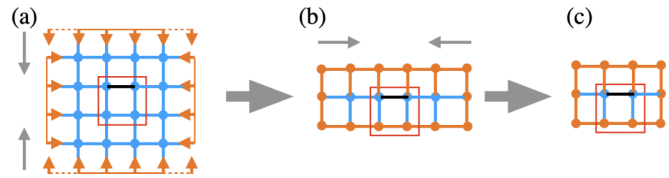

FIG. 6. Procedures to compute the local environment for the black bond inside a specific block taken from Fig. 1(e) of the main text. The standard bMPS algorithm for finite PEPS is used for this calculation.

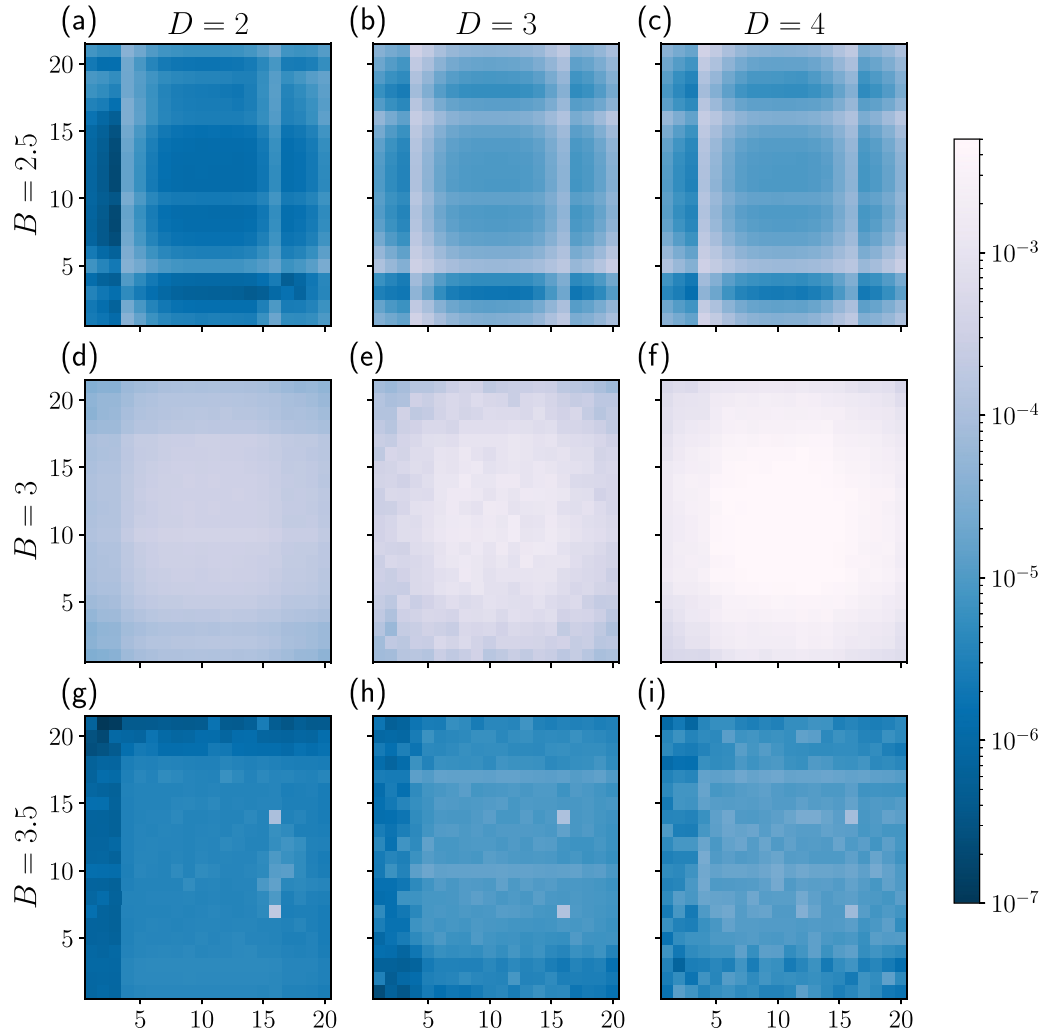

FIG. 7. Trace distances between the single site reduced density matrices of the horizontal bonds computed using bMPS and blockBP with block size 7×7. The ground states of the 21×21 transverse Ising model for different local fields B are used to compute the reduced density matrices. The columns from left to right are results for $D = 2, 3, 4$, respectively [i.e., (a), (d), (g) correspond to $D = 2$ and so on], while the rows from top to bottom are results for $B = 2.5, 3.0, 3.5$, respectively [i.e., (a)–(c) are for $B = 2.5$].

performing computations (either computing local observables or updating the site tensors) inside the center of each block. We use the bMPS [47] algorithm for both tasks, however, it is important to stress that in our case the bMPS algorithm is only applied to a small portion of the system, our blocks instead of the entire system, thus making our computations much more manageable and also parallelizable.

Let us consider, for example, the case shown in Fig. 1(d) of the main text. Here one wants to evaluate the message going from right to left from a certain block. We then need to contract the local tensor with the three incoming messages from the top, the right, and the bottom. Each of these messages is represented by an MPS. The contraction of the messages with the tensors in the block is done from right to left, column by column, each time keeping a bond dimension of (at most) $\chi_m = D^2$, where D is the virtual bond dimension of the tensors in the block. If the block we are considering is at the boundary of a finite system, then the MPS messages going into and out of it consist of only trivial tensors.

Once the messages have converged, we can compute an observable within a block or update the local site tensors inside a block during the ITE. Following Fig. 1(f) of the main text, we need to evaluate the contraction between the four converged incoming MPS messages, from top, bottom, left, and right with the tensors of the block (note that this can be straightforwardly generalized to other 2D lattices). Also, in this case one uses the bMPS algorithm on a single block plus messages, from any of the sides of the block. Since we aim to be more accurate when evaluating local observables at the center of the block, we use a larger bond dimension $\chi = 2D^2 + 10$ when doing these contractions.

The most computationally intensive subroutine involved in both tasks in the case of a square lattice is to multiply one line of three-legged tensors which represent an (updated) environment, with a line of four-legged tensors, e.g., the tensor of the block next to the environment. While this can be done by using the standard matrix product operator (MPO) time MPS arithmetic [45], and then compressing the resulting MPS, we

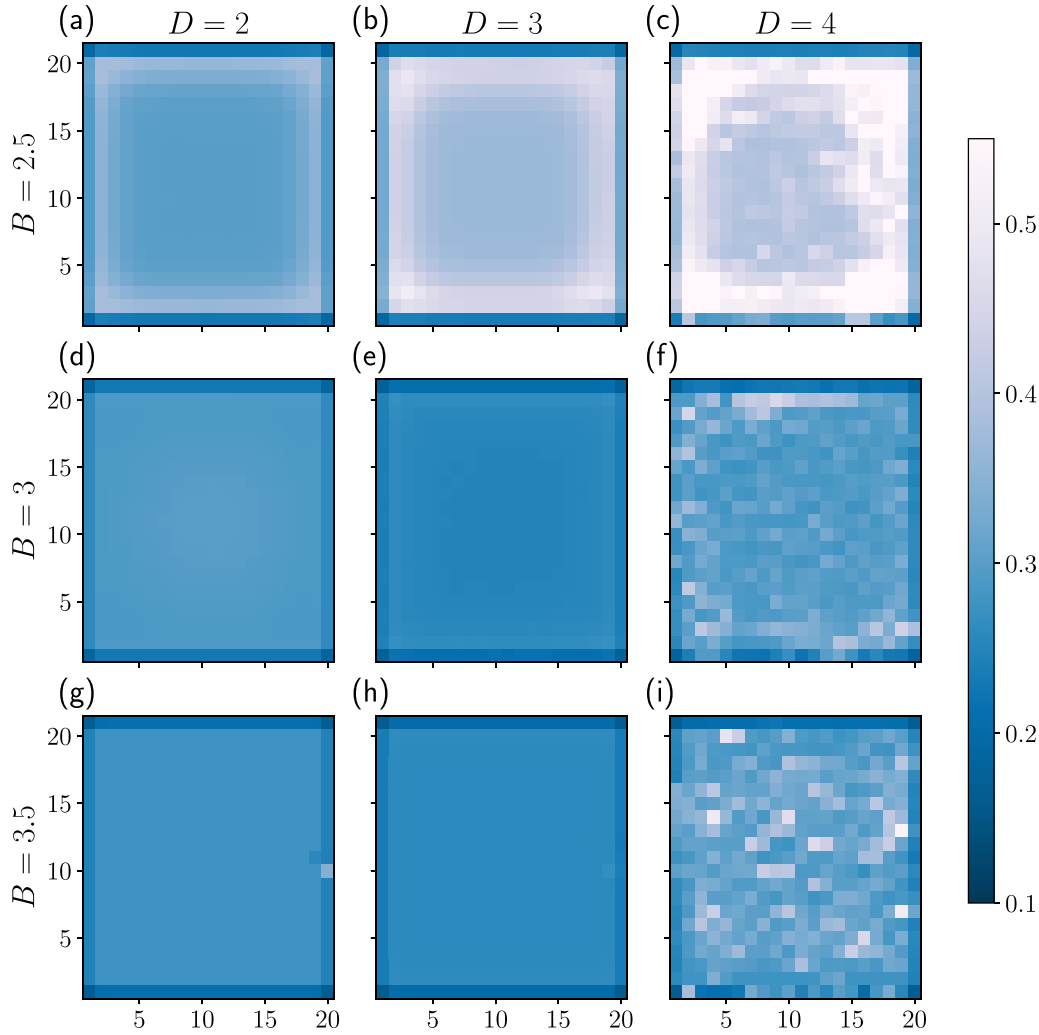

FIG. 8. Trace distances between the single site reduced density matrices of the horizontal bonds computed using bMPS and simple update. The other settings are the same as Fig. 7.

use the following zip-up algorithm that fuses these two operations into one to reduce both the memory and computational cost, as shown in Ref. [15].

We can write the input MPS $|X\rangle$ as

$$|X\rangle = \sum_{j=1}^{L} X^{a_1}_{b_1,b_2} X^{a_2}_{b_2,b_3} \cdots X^{a_L}_{b_L,b_{L+1}} \tag{A1}$$

and the input MPO $\hat{O}$ as

$$\hat{O} = \sum_{j=1}^{L} O^{a_1',a_1}_{c_1,c_2} O^{a_2',a_2}_{c_2,c_3} \cdots O^{a_L',a_L}_{c_L,c_{L+1}}. \tag{A2}$$

Our task is to calculate a fixed bond dimension MPS $|Y\rangle$, which approximates $\hat{O}|X\rangle$. We begin by first initializing $|Y\rangle$ randomly:

$$|Y\rangle = \sum_{j=1}^{L} Y^{a_1'}_{d_1,d_2} Y^{a_2'}_{d_2,d_3} \cdots Y^{a_L'}_{d_L,d_{L+1}}. \tag{A3}$$

Then we use an iterative algorithm to minimize the distance:

$$\begin{aligned} &\| |Y\rangle - \hat{O}|X\rangle \|^2 \\ &\quad = \langle Y|Y\rangle - \langle Y|\hat{O}|X\rangle - \langle X|\hat{O}^\dagger|Y\rangle + \langle X|\hat{O}^\dagger\hat{O}|X\rangle. \end{aligned} \tag{A4}$$

Note that the last term on the right-hand side of Eq. (A4) is a constant and can be neglected during the optimization. To optimize $|Y\rangle$, we iteratively update each site tensor of it using DMRG-like sweeps. For a specific site l, we first compute

$$\tilde{Y}^{a_l'}_{d_l,d_{l+1}} = \sum_{b_l,c_l,b_{l+1},c_{l+1},a_l} A^{d_l}_{c_l,b_l} O^{a_l',a_l}_{c_l,c_{l+1}} B^{d_{l+1}}_{c_{l+1},b_{l+1}} X^{a_l}_{b_l,b_{l+1}}, \tag{A5}$$

where the rank-3 tensors A and B can be computed iteratively using

$$A^{d_l}_{c_l,b_l} = \sum_{b_{l-1},c_{l-1},d_{l-1},a_{l-1},a_{l-1}'} A^{d_{l-1}}_{c_{l-1},b_{l-1}} Y^{a_{l-1}'}_{d_{l-1},d_l} O^{a_{l-1}',a_{l-1}}_{c_{l-1},c_l} X^{a_{l-1}}_{b_{l-1},b_l} \tag{A6}$$

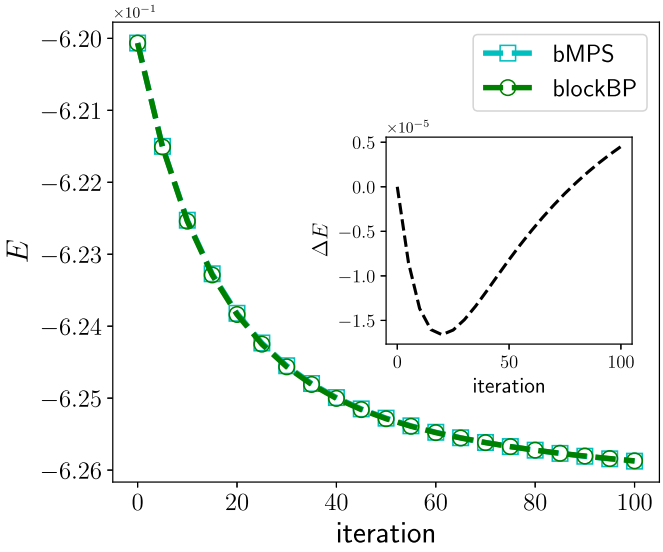

FIG. 9. The ground-state energy E obtained using bMPS (light blue squares) and blockBP (green circles) as a function of the imaginary time evolution steps. The inset shows the difference between the energies computed for the two algorithms, namely, $\Delta E = E_{\text{blockBP}} - E_{\text{bMPS}}$, the initial state for both cases is chosen as the ground state of the 10×10 AFH model with $D = 3$ and we have used $d\tau = 0.01$ in both cases.

and

$$B^{d_{l+1}}_{c_{l+1},b_{l+1}} = \sum_{b_{l+2},c_{l+2},d_{l+2},a_{l+1},a'_{l+1}} B^{d_{l+2}}_{c_{l+2},b_{l+2}} Y^{a'_{l+1}}_{d_{l+1},d_{l+2}} \times O^{a'_{l+1},a_{l+1}}_{c_{l+1},c_{l+2}} X^{a_{l+1}}_{b_{l+1},b_{l+2}}, \tag{A7}$$

with $A^{d_1}_{c_1,b_1} = B^{d_{L+1}}_{c_{L+1},b_{L+1}} = 1$. During the left to right sweep, we perform a QR decomposition (QR decomposition of a matrix M means that $M = QR$ where Q is a unitary matrix and R is an upper triangular matrix) of $\tilde{Y}^{a'_l}_{d_l,d_{l+1}}$ and get

$$\text{QR}(\tilde{Y}^{a'_l}_{d_l,d_{l+1}}) = \sum_s Q^{a'_l}_{d_l,s} R_{s,d_{l+1}}, \tag{A8}$$

and then we take $Q^{a'_l}_{d_l,s}$ as the new site tensor. During the right to left sweep, we perform an LQ decomposition (LQ decomposition of a matrix M means that $M = LQ$ where L is a lower triangular matrix and Q is a unitary matrix) of $\tilde{Y}^{a'_l}_{d_l,d_{l+1}}$ and get

$$\text{LQ}(\tilde{Y}^{a'_l}_{d_l,d_{l+1}}) = \sum_s L_{d_l,s} Q^{a'_l}_{s,d_{l+1}}, \tag{A9}$$

and then we take $Q^{a'_l}_{s,d_{l+1}}$ as the new site tensor. Equation (A5) is also demonstrated in Fig. 5. In addition, from Eq. (A5) we have $-\|\tilde{Y}^{a'_l}_{d_l,d_{l+1}}\|^2 = \| \, |Y\rangle - \hat{O}|X\rangle\|^2 - \langle X|\hat{O}^\dagger\hat{O}|X\rangle$, which is exactly the loss function subtracted by a constant term, therefore it can be used to monitor the convergence, similar to the ground-state energy in DMRG [45]. In our simulations, we have set the convergence criterion to be the standard deviation of $-\|\tilde{Y}^{a'_l}_{d_l,d_{l+1}}\|^2$ for a full sweep (left to right and then right to left) divided by the mean value, with a tolerance of 10^{-6} and the maximum number of sweeps to be 10.

For computing the MPS messages of a double layer TN, the MPS messages are randomly initialized as a matrix product density operator [59] using normal distribution, while for single-layer TN which is used for classical models (see the simulation of the classical Ising model below), the MPS messages are simply initialized as a random MPS using uniform distribution. Assuming that the mean square error between the MPS messages in the lth step and those in the $l-1$ step is ϵ_l, the convergence criterion in this case is chosen to be ϵ_l/ϵ_1, with a tolerance of 10^{-5} and the maximum number of iterations to be 10.

APPENDIX B: THE IMAGINARY TIME EVOLUTION ALGORITHM FOR PEPS

Generally, two different approaches are used to optimize the PEPS tensors for computing the ground state, namely, variational minimization and ITE. In this paper, we focus on the second approach. For a 2D Hamiltonian on a square lattice, written as $H = \sum_i h_i$, where i stands for all the nearest-neighbor pairs of sites (bonds) and h_i denotes the local Hamiltonian term on the ith bond, we first perform a first-order expansion of the imaginary time evolutionary operator $U(d\tau) = e^{-Hd\tau}$ for a small time interval $d\tau$ as

$$U(d\tau) \approx e^{-h_1 d\tau} e^{-h_2 d\tau} e^{-h_3 d\tau} \cdots . \tag{B1}$$

Each term on the right-hand side of Eq. (B1) acts on a specific bond, referred to as a gate. Changing the order of these terms on the right-hand side of Eq. (B1) still results in a first-order approximation, therefore one could freely adjust the order to maximally reuse the intermediate computations during the time evolution. In our implementation, we will first apply the gates on the horizontal bonds row by row, and then apply the gates on the vertical bonds column by column.

The application of a specific gate $G_i = e^{-h_i d\tau}$ is formulated as an optimization problem to minimize the loss function $\| |\psi'\rangle - G_i|\psi\rangle\|^2$, where $|\psi'\rangle$ is the new PEPS with all the tensors the same as $|\psi\rangle$ except the two tensors at bond i and $\|\cdot\|$ means the two-norm. Performing the local optimization requires us to calculate the environment around the tensors being updated, which is the central difference among various PEPS updating algorithms. In our approach, we first use blockBP to compute the boundary messages as MPSs for each block, which plays the role of the environment for each block. Then performing ITE on the tensors inside each block becomes almost the same as the original PEPS updating problem, except that we are dealing with a much smaller finite PEPS, and that there are four nontrivial MPSs on the boundaries (the standard finite PEPS update can also be taken as a special case where the boundaries are trivial MPSs for which the site tensors only contain a single element 1). As such, we can use existing finite PEPS update algorithms to update the tensors inside each block. In our implementation, we use the bMPS method algorithm inside each block, as demonstrated in Fig. 6 for applying a single gate on the black bond in a block taken from Fig. 1(e) of the main text. In Fig. 6(a), we map our problem to be exactly the same as the standard finite PEPS problem by padding four trivial tensors with single element 1 at the four corners. Then we treat the top and bottom rows of tensors as two MPSs, the middle rows of tensors as MPOs, and apply the MPOs onto the boundary MPSs (see Appendix A for the MPO-MPS multiplication) until there are

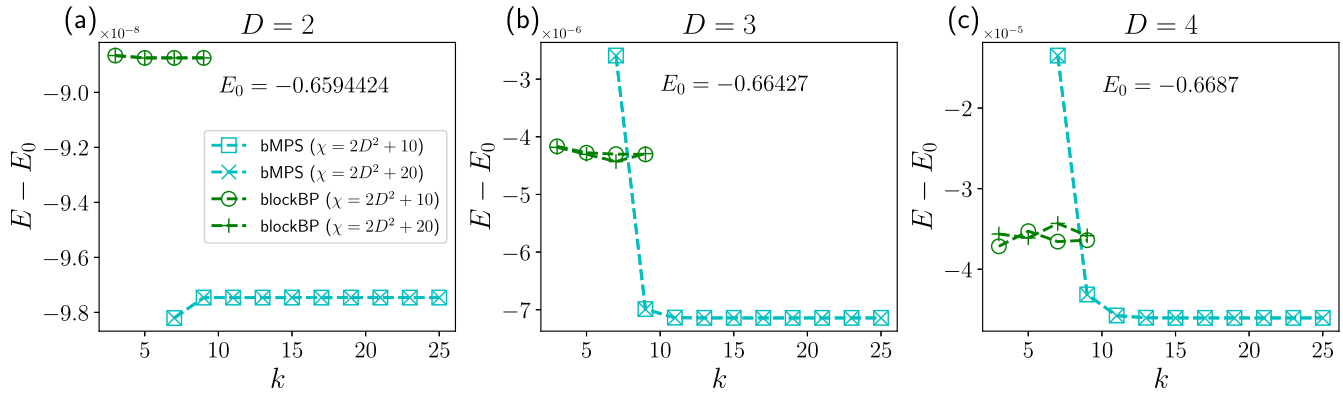

FIG. 10. Ground state energies of the infinite AFH model as a function of k computed using bMPS and blockBP for (a) $D = 2$, (b) $D = 3$ and (c) $D = 4$. For all those panels the cyan dashed lines with squares and x are energies computed using bMPS with $\chi = 2D^2 + 10$ and $\chi = 2D^2 + 20$ respectively, while the green dashed lines with circle and + are energies computed using blockBP with $\chi = 2D^2 + 10$ and $\chi = 2D^2 + 20$ respectively.

only three rows left, as shown in Fig. 6(b). After that, we perform tensor contractions from left and right until only the minimal environment for the two tensors is left, as shown in Fig. 6(c). Finally, we perform the alternating least-squares scheme, in which one of the two tensors will be updated at a time while keeping the other as constant, until the final loss converges. Overall, our implementation of bMPS update for each block closely follows the algorithm in Ref. [46].

APPENDIX C: EVALUATION OF REDUCED DENSITY MATRIX FOR FINITE SYSTEMS

Here we study the quality of the local environments in our blockBP by computing the two-site RDMs on each horizontal bond using blockBP and compare them to the RDMs computed using bMPS. The comparison is shown in the form of a heat map in Fig. 7, where we consider the ground state of a 21×21 transverse Ising model with OBC. The blockBP RDMs were computed using 7×7 blocks. Quantitatively, what we do is compute the reduced density matrix for the site in row i and columns j and $j+1$ with both methods, i.e., $\rho^{\text{bMPS}}_{i,j,j+1}$ and $\rho^{\text{blockBP}}_{i,j,j+1}$, and compute the trace distance between them, $d = \text{Tr}(|\rho^{\text{blockBP}}_{i,j,j+1} - \rho^{\text{bMPS}}_{i,j,j+1}|)/2$ with $|Z| = \sqrt{Z^\dagger Z}$. We can see that for $B = 2.5, 3.5$, which are away from the critical value ($B_c \approx 3.044$ [48]), the RDMs computed by these two methods are very close to each other with an average distance lower than 10^{-5} for all the bond dimensions D considered, while for $B = 3.0$ we get $d \approx 10^{-3}$ on average. This indicates that bMPS and blockBP lead to quantitative similar results, especially away from the transition point.

For comparison, in Fig. 8 we used the same setting as in Fig. 7, and calculated the trace distance between the RDMs computed using simple-update (or, in other words, the plain BP algorithm where the message MPS has only one leg) and the RDMs computed using bMPS. As can be seen from these heat maps, in these cases, the trace distance is of the order 10^{-1}, showing that blockBP environments are far superior to those of the plain BP or, equivalently, the simple-update algorithm.

APPENDIX D: CONVERGENCE OF BLOCKBP FOR FINITE SYSTEMS

In the main text, we benchmarked the ITE algorithm with the blockBP update against the same algorithm with the bMPS full update. We considered the AFMH model and looked at

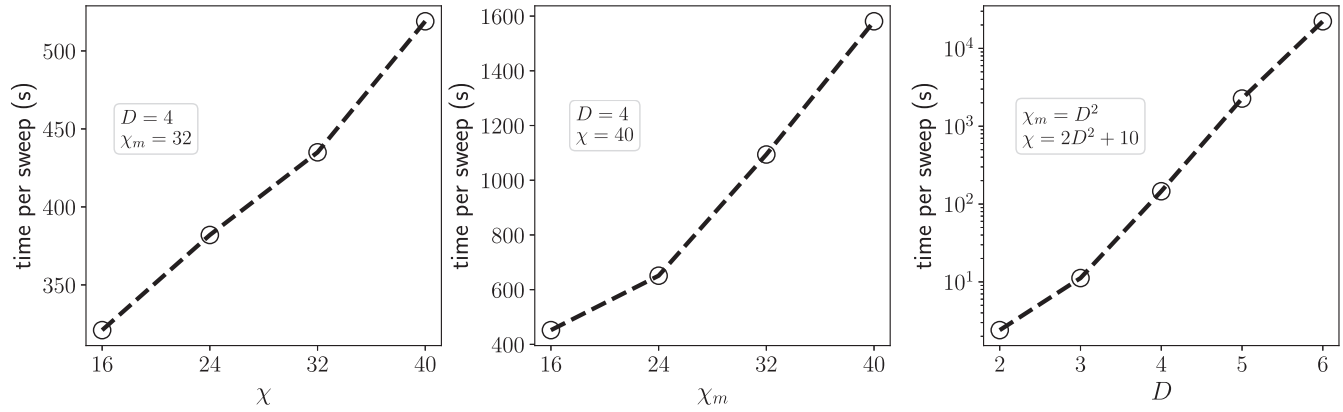

FIG. 11. Runtime per ITE sweep of blockBP for finite AFH model with size 14×14 against the hyperparameters χ (a), χ_m (b) and D (c). We have used a block size of 7×7. A single thread of the Intel(R) Core(TM) i7-11850H CPU (2.5 GHz) is used.

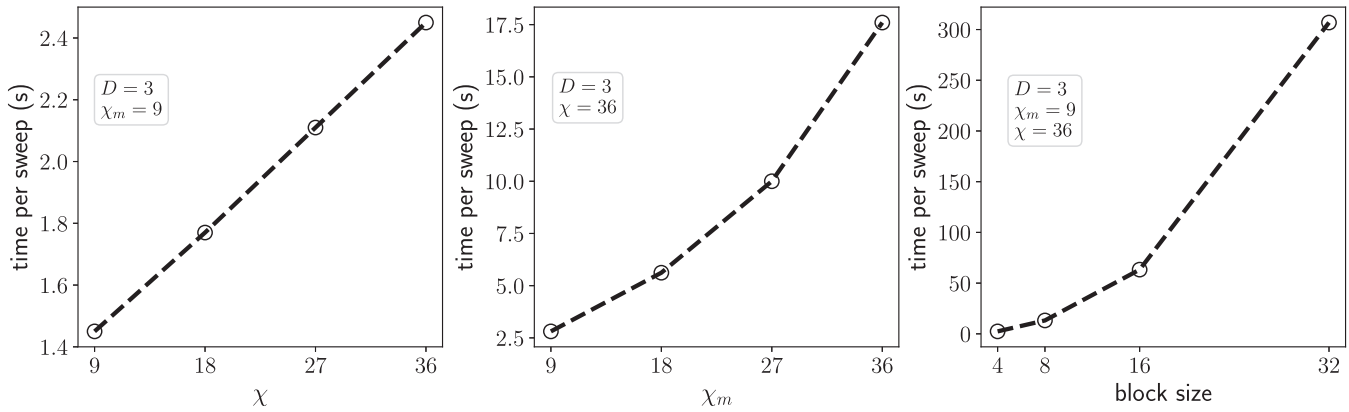

FIG. 12. Runtime per ITE sweep of blockBP for infinite transverse Ising model against the hyperparameters χ (a), χ_m (b), and block size (c). In (a), (b) we have used a block size of 4×4; we have also used $D = 3$ for all panels. A single thread of the Intel(R) Core(TM) i7-11850H CPU (2.5 GHz) is used.

the final result of the energy. Here we further demonstrate that even the evolution of our blockBP update towards the ground state has a very similar dynamics as bMPS, given the same parameters and the same initial state. This is due to the fact that both algorithms use the same Trotterized ITE [60] and they estimate in a very similar manner the Trotter steps. Specifically, in Fig. 9 we consider the ITE of a 10×10 AFH model using both bMPS and blockBP update (with block size 5×5) with $d\tau = 0.01$ and $D = 4$. For both algorithms, we took as the initial condition the ground state obtained using bMPS for $D = 3$. We can see from the inset that the difference between bMPS and blockBP is around 10^{-5} throughout the ITE.

APPENDIX E: COMPUTING LOCAL OBSERVABLES OF INFINITE SYSTEM USING BLOCKBP

In the following, we show the precision of our blockBP when used for computing local observables of an infinite system. Concretely, taking the infinite TN ground state computed in Fig. 4 in the main text using blockBP for different bond dimensions D, we compared two different ways of computing its energy. In the first approach, we embedded the TN with central block of size 2×2 into a finite system of size $(2k + 2) \times (2k + 2)$ where the boundaries are randomly initialized, and then we compute the energy in the 2×2 center using bMPS. Since the the system is infinite, by embedded we mean that the blocks are repeated to form the system. In the second approach, we calculated the ground energy using blockBP with a $2k \times 2k$ block size. In this case, the boundary of the system is given by the MPS messages, while for bMPS we have used a random MPS. The results are shown in Fig. 10. Given the high accuracy of the results, we show both for blockBP and bMPS the energy minus an offset value of E_0. We see that both approaches converge to the fifth digit for large enough k, but that the blockBP results converge much faster—already with a small $k = 3$. Therefore, blockBP can also be used as an efficient method to accurately compute local observables in infinite systems. Figure 10 also shows the ground-state energies computed using a larger $\chi = 2\mathrm{D}^2 + 20$ and, as we can see, they are very close to the energies computed using $\chi = 2\mathrm{D}^2 + 10$ (the latter is chosen as the default throughout this paper).

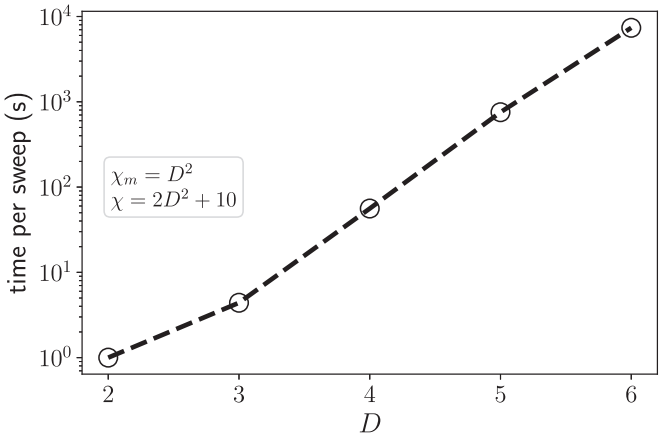

FIG. 13. Runtime for five ITE sweeps of blockBP for infinite AFH model against D. We have used a block size of 4×4. Eight threads of the Intel(R) Core(TM) i7-11850H CPU (2.5 GHz) are used.

APPENDIX F: RUNTIME SCALING OF BLOCKBP AGAINST HYPERPARAMETERS

Here we show more details of the runtime scaling of our blockBP versus the four hyperparameters, namely, the bond dimension D of the PEPS, the bond dimension χ used during bMPS update, the bond dimension χ_m used for computing the MPS messages, and the block size. Specifically, in Fig. 11, we show the runtime scaling of blockBP for the finite AFH model against χ, χ_m, and D with block size fixed to be 7×7. In Fig. 12, we show the runtime scaling of blockBP for the infinite TI model against χ, χ_m, and block size with D fixed to be 3. In Fig. 13, we show the runtime scaling of blockBP for the infinite AFH model against D, with χ and χ_m using their default values as in the main text. From Figs. 11 and 12, we can see that the runtime scales extremely rapidly with D, and scales slightly faster than linear against χ_m and the block size, while in comparison the scaling against χ is a lot slower

than χ_m. The latter is due to the fact that when computing messages, we have used several iterations, while during the bMPS update performed in each block, only two iterations are required (one horizontal and one vertical). Nevertheless, as shown in the main text, the default value of $\chi_m = D^2$ could already result in accurate results in all the cases considered in this paper. In Fig. 13, we show the runtime scaling of the infinite AFH model against D, with χ and χ_m set to their default values. We note that the runtime scaling of a fast full update algorithm has been studied in Fig. 13 of Ref. [50], where we can see that for $D \leqslant 5$ we are faster while for $D = 6$ we are slower. However, we stress that this comparison is not exact since the computational resources used for Fig. 13 of Ref. [50] are not shown in that paper.

[1] W. M. C. Foulkes, L. Mitas, R. J. Needs, and G. Rajagopal, Rev. Mod. Phys. **73**, 33 (2001).
[2] R. Orús, Ann. Phys. **349**, 117 (2014).
[3] M. Troyer and U.-J. Wiese, Phys. Rev. Lett. **94**, 170201 (2005).
[4] F. Verstraete and J. I. Cirac, arXiv:cond-mat/0407066.
[5] R. Orús, Nat. Rev. Phys. **1**, 538 (2019).
[6] J. I. Cirac, D. Pérez-García, N. Schuch, and F. Verstraete, Rev. Mod. Phys. **93**, 045003 (2021).
[7] P. Corboz, T. M. Rice, and M. Troyer, Phys. Rev. Lett. **113**, 046402 (2014).
[8] P. Corboz and F. Mila, Phys. Rev. B **87**, 115144 (2013).
[9] P. Corboz and F. Mila, Phys. Rev. Lett. **112**, 147203 (2014).
[10] B.-X. Zheng, C.-M. Chung, P. Corboz, G. Ehlers, M.-P. Qin, R. M. Noack, H. Shi, S. R. White, S. Zhang, and G. K.-L. Chan, Science **358**, 1155 (2017).
[11] H. J. Liao, Z. Y. Xie, J. Chen, Z. Y. Liu, H. D. Xie, R. Z. Huang, B. Normand, and T. Xiang, Phys. Rev. Lett. **118**, 137202 (2017).
[12] Q. Li, H. Li, J. Zhao, H.-G. Luo, and Z. Y. Xie, Phys. Rev. B **105**, 184418 (2022).
[13] Z. Y. Xie, J. Chen, M. P. Qin, J. W. Zhu, L. P. Yang, and T. Xiang, Phys. Rev. B **86**, 045139 (2012).
[14] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac, Phys. Rev. Lett. **98**, 140506 (2007).
[15] F. Verstraete, V. Murg, and J. Cirac, Adv. Phys. **57**, 143 (2008).
[16] T. Nishino and K. Okunishi, J. Phys. Soc. Jpn. **65**, 891 (1996).
[17] R. Orús and G. Vidal, Phys. Rev. B **80**, 094403 (2009).
[18] A. W. Sandvik and G. Vidal, Phys. Rev. Lett. **99**, 220602 (2007).
[19] L. Wang, I. Pižorn, and F. Verstraete, Phys. Rev. B **83**, 134421 (2011).
[20] M. Levin and C. P. Nave, Phys. Rev. Lett. **99**, 120601 (2007).
[21] H. C. Jiang, Z. Y. Weng, and T. Xiang, Phys. Rev. Lett. **101**, 090603 (2008).
[22] G. Evenbly and G. Vidal, Phys. Rev. Lett. **115**, 180405 (2015).
[23] M. J. Wainwright and M. I. Jordan, Found. Trends Mach. Learn. **1**, 1 (2008).
[24] J. Pearl, *Proceedings of the Second AAAI Conference on Artificial Intelligence*, *AAAI'82* (AAAI Press, Palo Alto, California, 1982), pp. 133–136.
[25] M. Mézard, G. Parisi, and M. A. Virasoro, *Spin Glass Theory and Beyond: An Introduction to the Replica Method and Its Applications* (World Scientific Publishing Company, Singapore, 1987), Vol. 9.
[26] S. Yoon, A. V. Goltsev, S. N. Dorogovtsev, and J. F. F. Mendes, Phys. Rev. E **84**, 041144 (2011).
[27] B. Karrer, M. E. J. Newman, and L. Zdeborová, Phys. Rev. Lett. **113**, 208702 (2014).
[28] M. Mézard, G. Parisi, and R. Zecchina, Science **297**, 812 (2002).
[29] M. Mezard and A. Montanari, *Information, Physics, and Computation* (Oxford University Press, Walton Street, Oxford, 2009).
[30] B. Karrer and M. E. J. Newman, Phys. Rev. E **82**, 016101 (2010).
[31] J. S. Yedidia, W. Freeman, and Y. Weiss, in *Advances in Neural Information Processing Systems*, edited by T. Leen, T. Dietterich, and V. Tresp (MIT Press, Cambridge, Massachusetts, 2000), Vol. 13.
[32] J. Yedidia, W. Freeman, and Y. Weiss, IEEE Trans. Inf. Theory **51**, 2282 (2005).
[33] A. Pelizzola, J. Phys. A: Math. Gen. **38**, R309 (2005).
[34] A. Montanari and T. Rizzo, J. Stat. Mech.: Theory Exp. (2005) P10011.
[35] M. Chertkov and V. Y. Chernyak, J. Stat. Mech.: Theory Exp. (2006) P06009.
[36] J. Mooij, B. Wemmenhove, B. Kappen, and T. Rizzo, *Artificial Intelligence and Statistics* (PMLR, Corvalis, Oregon, USA, 2007), pp. 331–338.
[37] A. Lage-Castellanos, R. Mulet, F. Ricci-Tersenghi, and T. Rizzo, J. Phys. A: Math. Theor. **46**, 135001 (2013).
[38] C. Wang and H.-J. Zhou, J. Phys.: Conf. Ser. **473**, 012004 (2013).
[39] H.-J. Zhou and W.-M. Zheng, Eur. Phys. J. B **88**, 336 (2015).
[40] G. T. Cantwell and M. E. J. Newman, Proc. Natl. Acad. Sci. USA **116**, 23398 (2019).
[41] A. Kirkley, G. T. Cantwell, and M. E. J. Newman, Sci. Adv. **7**, eabf1211 (2021).
[42] R. Alkabetz and I. Arad, Phys. Rev. Res. **3**, 023073 (2021).
[43] S. Sahu and B. Swingle, arXiv:2206.04701.
[44] T. Helgaker, P. Jorgensen, and J. Olsen, Front matter, in *Molecular Electronic-Structure Theory* (John Wiley & Sons, Ltd., Hoboken, New Jersey, 2000), Chap. 5, pp. 142–196.
[45] U. Schollwöck, Ann. Phys. **326**, 96 (2011).
[46] M. Lubasch, J. I. Cirac, and M.-C. Bañuls, New J. Phys. **16**, 033014 (2014).
[47] M. Lubasch, J. I. Cirac, and M.-C. Bañuls, Phys. Rev. B **90**, 064425 (2014).
[48] H. W. J. Blöte and Y. Deng, Phys. Rev. E **66**, 066110 (2002).
[49] A. W. Sandvik, Phys. Rev. B **56**, 11678 (1997).
[50] H. N. Phien, J. A. Bengua, H. D. Tuan, P. Corboz, and R. Orús, Phys. Rev. B **92**, 035142 (2015).
[51] L. Onsager, Phys. Rev. **65**, 117 (1944).
[52] C. N. Yang, Phys. Rev. **85**, 808 (1952).
[53] Z. Y. Xie, H. C. Jiang, Q. N. Chen, Z. Y. Weng, and T. Xiang, Phys. Rev. Lett. **103**, 160601 (2009).

[54] H. H. Zhao, Z. Y. Xie, Q. N. Chen, Z. C. Wei, J. W. Cai, and T. Xiang, Phys. Rev. B **81**, 174411 (2010).
[55] H. Zhou and C. Wang, J. Stat. Phys. **148**, 513 (2012).
[56] M. Scheb and R. M. Noack, Phys. Rev. B **107**, 165112 (2023).
[57] P. Corboz, Phys. Rev. B **94**, 035133 (2016).
[58] H.-J. Liao, J.-G. Liu, L. Wang, and T. Xiang, Phys. Rev. X **9**, 031041 (2019).
[59] F. Verstraete, J. J. García-Ripoll, and J. I. Cirac, Phys. Rev. Lett. **93**, 207204 (2004).
[60] N. Hatano and M. Suzuki, Finding exponential product formulas of higher orders, in *Quantum Annealing and Other Optimization Methods*, edited by A. Das and B. K. Chakrabarti (Springer, Berlin, 2005), pp. 37–68.