Sublattice coding algorithm and distributed memory parallelization for large-scale exact diagonalizations of quantum many-body systems

Alexander Wietek* and Andreas M. Läuchli
Institut für Theoretische Physik, Universität Innsbruck, A-6020 Innsbruck, Austria

(Received 18 April 2018; published 26 September 2018)

We present algorithmic improvements for fast and memory-efficient use of discrete spatial symmetries in exact diagonalization computations of quantum many-body systems. These techniques allow us to work flexibly in the reduced basis of symmetry-adapted wave functions. Moreover, a parallelization scheme for the Hamiltonian-vector multiplication in the Lanczos procedure for distributed memory machines avoiding load-balancing problems is proposed. We demonstrate that using these methods low-energy properties of systems of up to 50 spin-1/2 particles can be successfully determined.

DOI: 10.1103/PhysRevE.98.033309

I. INTRODUCTION

Exact diagonalization (ED) studies have in the past been a reliable source of numerical insight into various problems in quantum many-body physics, ranging from quantum chemistry [1], nuclear structure [2–4], and quantum field theory [5] to strongly correlated lattice models in condensed matter physics. The method is versatile, unbiased, and capable of simulating systems with a sign problem. The main limitation of ED is the typically exponential scaling of computational effort and memory requirements in the system size. Nevertheless, the number of particles or lattice sites feasible for simulation has steadily increased since the early beginnings [6] and have provided valuable insight into many problems in modern condensed matter physics, for example, frustrated magnetism [7–14], high-temperature superconductivity [15–18], the quantum Hall effect and fractional Chern insulators [19–22], and quantum critical points in 2+1 dimensions [23,24]. Different approaches for increasing the system size in these simulations have been proposed [25,26]. Not only does increasing the number of particles yield better approximations to the thermodynamic limit, but also several interesting simulation clusters with many symmetries become available if more particles can be simulated. Having access to such clusters becomes important if several competing phases ought to be realized on the same finite-size sample.

The ED method is essentially equivalent to simulating quantum circuits. With the advent of scalable experimental quantum computation [27–29], exact classical simulation of quantum circuits has become important for benchmarking and validating results from actual quantum computers [30–32]. At present, we are on the verge of quantum computers surpassing the capabilities of classical supercomputers in terms of the number of simulated qubits, colloquially referred to as quantum advantage. Specifically, the barrier of classically simulating 50 qubits has not been breached to date.

In this work, we present algorithms and strategies for the implementation of a state-of-the-art large-scale ED code and prove that applying these methods systems of up to 50 spin-1/2 particles can be simulated on present-day supercomputers. Two key ingredients make these computations possible:

(1) *Efficient use of symmetries*. We present an algorithm to work with symmetry-adapted wave functions in a fast and memory efficient way. This so-called *sublattice coding algorithm* allows us to diagonalize the Hamiltonian in every irreducible representation of a discrete symmetry group. The basic idea behind this algorithm goes back to Lin [25]. An extension of this method was proposed in Ref. [26]. We generalize these approaches to arbitrary discrete symmetries, varying the number of sublattices and arbitrary geometries.

(2) *Parallelization of the matrix-vector multiplications* in the Lanczos algorithm [33] for distributed memory machines. We propose a method avoiding load-balancing problems in message passing and present a computationally fast way of storing the Hilbert space basis.

These ideas have been implemented and tested on various supercomputers. We present results and benchmarks to demonstrate the efficiency and flexibility of the proposed methods.

II. SYMMETRY ADAPTED BASIS STATES

Employing symmetries in ED computations amounts to block diagonalizing the Hamiltonian. The blocks correspond to the irreducible representations of the symmetry group and the procedure of block diagonalization amounts to changing the basis of the Hilbert space to *symmetry-adapted basis states*. Here we briefly review this basis and recall some basic notions commonly used in this context. For a more detailed introduction to this topic see, e.g., Refs. [34,35]. In this paper, we consider only one-dimensional representations of the symmetry group. Consider a generic spin configuration on N lattice sites with local dimension d,

$$|\boldsymbol{\sigma}\rangle = |\sigma_1, \ldots, \sigma_N\rangle, \quad \sigma_i \in \{1, \ldots, d\}. \tag{1}$$

*awietek@flatironinstitute.org

The symmetry-adapted basis states $|\boldsymbol{\sigma}_\rho\rangle$ are given by

$$|\boldsymbol{\sigma}_\rho\rangle \equiv \frac{1}{N_{\rho,\boldsymbol{\sigma}}} \sum_{g\in\mathcal{G}} \chi_\rho(g)^* g|\boldsymbol{\sigma}\rangle, \tag{2}$$

where $\mathcal{G}$ denotes a discrete symmetry group, ρ a one-dimensional representation of this group, $\chi_\rho(g)$ the character of this representation evaluated at group element g, and $N_{\rho,\boldsymbol{\sigma}}$ the normalization constant of the state $|\boldsymbol{\sigma}_\rho\rangle$. The set of basis state spin configurations $|\boldsymbol{\sigma}\rangle$ is divided into *orbits*,

$$\mathrm{Orbit}(|\boldsymbol{\sigma}\rangle) = \{g|\boldsymbol{\sigma}\rangle | g \in \mathcal{G}\}. \tag{3}$$

We define

$$|\boldsymbol{\sigma}\rangle < |\boldsymbol{\sigma}'\rangle :\Leftrightarrow \mathrm{int}(|\boldsymbol{\sigma}\rangle) < \mathrm{int}(|\boldsymbol{\sigma}'\rangle), \tag{4}$$

where $\mathrm{int}(|\boldsymbol{\sigma}\rangle)$ denotes an integer value coding on the computer for the spin configuration $|\boldsymbol{\sigma}\rangle$. The *representative* $|\tilde{\boldsymbol{\sigma}}\rangle$ within each orbit is given by the element with smallest integer value,

$$|\tilde{\boldsymbol{\sigma}}\rangle = g_{\boldsymbol{\sigma}}|\boldsymbol{\sigma}\rangle, \quad g_{\boldsymbol{\sigma}} = \underset{g\in\mathcal{G}}{\mathrm{argmin}}\ \mathrm{int}(g|\boldsymbol{\sigma}\rangle). \tag{5}$$

The matrix element $\langle\tilde{\boldsymbol{\sigma}'}_\rho|H_k|\tilde{\boldsymbol{\sigma}}_\rho\rangle$ for nonbranching terms H_k for two symmetry-adapted basis states with representation ρ is given by

$$\langle\tilde{\boldsymbol{\sigma}'}_\rho|H_k|\tilde{\boldsymbol{\sigma}}_\rho\rangle = \chi_\rho(g_{\boldsymbol{\sigma}'})\frac{N_{\rho,\boldsymbol{\sigma}'}}{N_{\rho,\boldsymbol{\sigma}}}\langle\boldsymbol{\sigma}'|H_k|\tilde{\boldsymbol{\sigma}}\rangle. \tag{6}$$

III. SUBLATTICE CODING ALGORITHM

Evaluating the matrix elements $\langle\tilde{\boldsymbol{\sigma}'}_\rho|H_k|\tilde{\boldsymbol{\sigma}}_\rho\rangle$ in Eq. (6) for all basis states $|\tilde{\boldsymbol{\sigma}}_\rho\rangle$ and $|\tilde{\boldsymbol{\sigma}'}_\rho\rangle$ efficiently is the gist of employing symmetries in ED computations. In an actual implementation on the computer we need to perform the following steps:

(1) Apply the nonbranching term H_k on the representative state $|\tilde{\boldsymbol{\sigma}}\rangle$. This yields a possibly nonrepresentative state $|\boldsymbol{\sigma}'\rangle$. From this, we can compute the factor $\langle\boldsymbol{\sigma}'|H_k|\tilde{\boldsymbol{\sigma}}\rangle$.

(2) Find the representative $|\tilde{\boldsymbol{\sigma}'}\rangle$ of $|\boldsymbol{\sigma}'\rangle$ and determine the group element $g_{\boldsymbol{\sigma}'}$ such that $|\tilde{\boldsymbol{\sigma}'}\rangle = g_{\boldsymbol{\sigma}'}|\boldsymbol{\sigma}'\rangle$. This yields the factor $\chi_\rho(g_{\boldsymbol{\sigma}'})$.

(3) Know the normalization constants $N_{\rho,\boldsymbol{\sigma}'}$ and $N_{\rho,\boldsymbol{\sigma}}$. These are usually computed when creating a list of all representatives and stored in a separate list.

The problem of finding the representative $|\tilde{\boldsymbol{\sigma}}\rangle$ of a given state $|\boldsymbol{\sigma}\rangle$ and its corresponding symmetry $g_{\boldsymbol{\sigma}}$ turns out to be the computational bottleneck of ED in a symmetrized basis. It is thus desirable to solve this problem quickly and in a memory-efficient manner. There are two straightforward approaches to solving this problem:

(1) Apply all symmetries directly to $|\boldsymbol{\sigma}'\rangle$ to find the minimizing group element $g_{\boldsymbol{\sigma}'}$,

$$g_{\boldsymbol{\sigma}'} = \underset{g\in\mathcal{G}}{\mathrm{argmin}}\ \mathrm{int}(g|\boldsymbol{\sigma}'\rangle). \tag{7}$$

This method does not have any memory overhead but is computationally slow since all symmetries have to be applied to the given state $|\boldsymbol{\sigma}'\rangle$.

(2) For every state $|\boldsymbol{\sigma}\rangle$ we store $|\tilde{\boldsymbol{\sigma}}\rangle$ and $g_{\boldsymbol{\sigma}}$ in a lookup table. While this is very fast computationally, the lookup table for storing all representatives grows exponentially in the system size.

The key to solving the representative search problem adequately is to have an algorithm that is almost as fast as a lookup table, where memory requirements are within reasonable bounds. This problem has already been addressed by several authors [25,26]. The central idea in these so-called *sublattice coding techniques* is to have a lookup table for the representatives on a sublattice of the original lattice and combine the information of the sublattice representatives to compute the total representative. These ideas were first introduced in Refs. [11,25,26]. In the following paragraphs, we explain the basic idea behind these algorithms and propose a flexible extension to arbitrary geometries and number of sublattices.

A. Sublattice coding on two sublattices

For demonstration purposes, we consider a simple translationally invariant spin-1/2 system on a six-site chain lattice with periodic boundary conditions. The lattice is divided into two sublattices as in Fig. 1. The even sites form sublattice A, and the odd sites form sublattice B. We enumerate the sites such that sites 1 to 3 are in sublattice A and sites 4 to 6 are in sublattice B. We choose the integer representation of a state $|\boldsymbol{\sigma}\rangle$ such that the most significant bits are formed by the spins in sublattice A. The symmetry group we consider consists of the six translations on the chain

$$\mathcal{G} = \{\mathrm{Id}, T, T_2, T_3, T_4, T_5\}, \tag{8}$$

where T_n denotes the translation by n lattice sites. The splitting of the lattice into two sublattices is stable in the sense that every symmetry element $g \in \mathcal{G}$ either maps the A sublattice to A and the B sublattice to B or the A sublattice to B and the B sublattice to A. We call this property *sublattice stability*. It is both a property of the partition of our lattice into sublattices and the symmetry group. Hence, the symmetry group is composed of two kinds of symmetries:

$$\begin{aligned} \mathcal{G}_A &\equiv \{g \in \mathcal{G};\quad g \text{ maps sublattice } A \text{ onto } A\},\\ \mathcal{G}_B &\equiv \{g \in \mathcal{G};\quad g \text{ maps sublattice } B \text{ onto } A\}. \end{aligned} \tag{9}$$

We denote by $|\boldsymbol{\sigma}\rangle_A$ (resp. $|\boldsymbol{\sigma}\rangle_B$) the state restricted to sublattice A (resp. B) and define the *sublattice representatives*,

$$\begin{aligned} \mathtt{Rep}_A(|\boldsymbol{\sigma}\rangle_A) &\equiv h_A|\boldsymbol{\sigma}\rangle_A, \quad h_A = \underset{g\in\mathcal{G}_A}{\mathrm{argmin}}\ \mathrm{int}(g|\boldsymbol{\sigma}\rangle_A),\\ \mathtt{Rep}_B(|\boldsymbol{\sigma}\rangle_B) &\equiv h_B|\boldsymbol{\sigma}\rangle_B, \quad h_B = \underset{g\in\mathcal{G}_B}{\mathrm{argmin}}\ \mathrm{int}(g|\boldsymbol{\sigma}\rangle_B), \end{aligned} \tag{10}$$

and the *representative symmetries*,

$$\begin{aligned} \mathtt{Sym}_A(|\boldsymbol{\sigma}\rangle_A) &\equiv \{g \in \mathcal{G}_A;\quad g|\boldsymbol{\sigma}\rangle_A = \mathtt{Rep}_A(|\boldsymbol{\sigma}\rangle_A)\},\\ \mathtt{Sym}_B(|\boldsymbol{\sigma}\rangle_B) &\equiv \{g \in \mathcal{G}_B;\quad g|\boldsymbol{\sigma}\rangle_B = \mathtt{Rep}_B(|\boldsymbol{\sigma}\rangle_B)\}. \end{aligned} \tag{11}$$

Let again $|\tilde{\boldsymbol{\sigma}}\rangle = g_{\boldsymbol{\sigma}}|\boldsymbol{\sigma}\rangle$, where $|\tilde{\boldsymbol{\sigma}}\rangle$ is the representative of $|\boldsymbol{\sigma}\rangle$. The minimizing symmetry $g_{\boldsymbol{\sigma}}$ can be an element of $\mathtt{Sym}_A(|\boldsymbol{\sigma}\rangle_A)$ only if $\mathtt{Rep}_A(|\boldsymbol{\sigma}\rangle_A) \leqslant \mathtt{Rep}_B(|\boldsymbol{\sigma}\rangle_B)$ or vice versa.

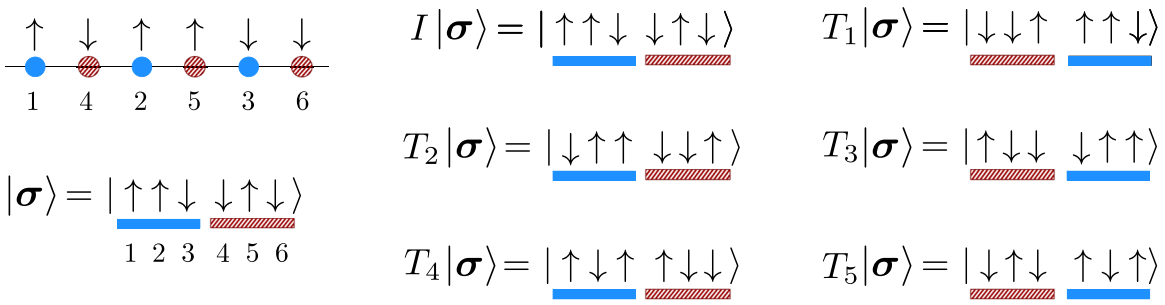

FIG. 1. Two sublattices coding of the spin state $|\sigma\rangle$ on a six-site chain lattice and action of translational symmetries. The sites are enumerated such that sites 1–3 are on the blue (solid) sublattice A, 4–6 on the red (dashed) sublattice B. The representative state with this enumeration of sites is given by $|\tilde{\sigma}\rangle = T_1|\sigma\rangle = |\downarrow\downarrow\uparrow\uparrow\uparrow\downarrow\rangle$. Notice that the symmetries act on real space, and thus the transformation of the basis states also depends on the numbering of sites.

Put differently,

$$\mathtt{Rep}_B(|\sigma\rangle_B) < \mathtt{Rep}_A(|\sigma\rangle_A) \Rightarrow g_\sigma \notin \mathtt{Sym}_A(|\sigma\rangle_A). \tag{12}$$

Otherwise, any symmetry element in $\mathtt{Rep}_B(|\sigma\rangle_B)$ would yield a smaller integer value than g_σ. This is the core idea behind the sublattice coding technique. We store $\mathtt{Rep}_{A,B}(|\sigma\rangle_{A,B})$ for every substate $|\sigma\rangle_{A,B}$ in a lookup table together with $\mathtt{Sym}_{A,B}(|\sigma\rangle_{A,B})$. In a first step, we determine the sublattice representative with smallest most significant bits. Then we apply the representative symmetries to $|\sigma\rangle$ in order to determine the true representative $|\tilde{\sigma}\rangle$. The number of representative symmetries $|\mathtt{Sym}_{A,B}(|\sigma\rangle_{A,B})|$ is typically much smaller than the total number of symmetries $|\mathcal{G}|$. The following example illustrates the idea and shows how to compute the representative given the information about sublattice representatives and representative symmetries.

Example. We consider the state $|\sigma\rangle = |\uparrow\uparrow\downarrow\downarrow\uparrow\downarrow\rangle$ on a six-site chain lattice as in Fig. 1. Notice that the sites are not enumerated from left to right but such that sites 1 to 3 belong to the sublattice A and sites 4 to 6 belong to sublattice B. The states restricted on the sublattices are $|\sigma\rangle_A = |\uparrow\uparrow\downarrow\rangle$ and $|\sigma\rangle_B = |\downarrow\uparrow\downarrow\rangle$. The action of the sublattice symmetries

$$\begin{aligned} \mathcal{G}_A &\equiv \{\mathrm{Id}, T_2, T_4\}, \\ \mathcal{G}_B &\equiv \{T_1, T_3, T_5\}, \end{aligned} \tag{13}$$

on $|\sigma\rangle$ is shown in Fig. 1. From this, we compute the sublattice representatives as in Eq. (10),

$$\begin{aligned} \mathtt{Rep}_A(|\sigma\rangle_A) &= |\downarrow\uparrow\uparrow\rangle, \\ \mathtt{Rep}_B(|\sigma\rangle_B) &= |\downarrow\downarrow\uparrow\rangle, \end{aligned} \tag{14}$$

whose integer values are given by

$$\begin{aligned} \mathrm{int}(\mathtt{Rep}_A(|\sigma\rangle_A)) &= (011)_2 = 3, \\ \mathrm{int}(\mathtt{Rep}_B(|\sigma\rangle_B)) &= (001)_2 = 1. \end{aligned} \tag{15}$$

Since $\mathtt{Rep}_B(|\sigma\rangle_B) < \mathtt{Rep}_A(|\sigma\rangle_A)$ the symmetry g_σ yielding the total representative $|\tilde{\sigma}\rangle$ must be contained in

$$\mathtt{Sym}_B(|\sigma\rangle_B) = \{T_1\}, \tag{16}$$

which in this case just contains a single element: T_1. Consequently, the representative $|\tilde{\sigma}\rangle$ is given by

$$|\tilde{\sigma}\rangle = T_1|\sigma\rangle = |\downarrow\downarrow\uparrow\uparrow\uparrow\downarrow\rangle. \tag{17}$$

Lookup tables. If the quantities $\mathtt{Rep}_{A,B}(|\sigma\rangle_{A,B})$ and $\mathtt{Sym}_{A,B}(|\sigma\rangle_{A,B})$ are now stored in a lookup table, this computation can be done very efficiently. Notice that instead of having to store 2^N entries in the lookup table for the representative we need only four lookup tables of order $O(2^{N/2})$, two for the quantities $\mathtt{Rep}_A(|\sigma\rangle_A)$ and $\mathtt{Rep}_B(|\sigma\rangle_B)$, and two for $\mathtt{Sym}_A(|\sigma\rangle_A)$ and $\mathtt{Sym}_B(|\sigma\rangle_B)$. On larger system sizes the difference between memory requirements of order $O(2^N)$ and $O(2^{N/2})$ is substantial.

To further speed up computations we also create lookup tables to store the action of each symmetry $g \in \mathcal{G}$ on a substate $|\sigma\rangle_A$,

$$\begin{aligned} \mathtt{SymmetryAction}_A(g, |\sigma\rangle_A) &= g|\sigma\rangle_A, \\ \mathtt{SymmetryAction}_B(g, |\sigma\rangle_B) &= g|\sigma\rangle_B. \end{aligned} \tag{18}$$

With this information, we can efficiently apply symmetries to a given spin configuration by looking up the action of g on the respective substate and combining the results. The memory requirement for these lookup tables is $O(N_{\mathrm{sym}}2^{N/2})$, where $N_{\mathrm{sym}} = |\mathcal{G}|$. This can be reduced by generalizing the sublattice coding algorithm to multiple sublattices, as explained in the following section. In that case, $2N_{\mathrm{sublat}}$ lookup tables of size $O(2^{N/N_{\mathrm{sublat}}})$ are required for storing the sublattice representatives $\mathtt{Rep}_X(|\sigma\rangle_X)$ and representative symmetries $\mathtt{Sym}_X(|\sigma\rangle_X)$, as defined in Eqs. (21) and (22), respectively. N_{sublat} denotes the number of sublattices. For storing the action of each symmetry $\mathtt{SymmetryAction}_X(g, |\sigma\rangle_X)$ as in Eq. (23) we further need N_{sublat} lookup tables of size $O(N_{\mathrm{sym}}2^{N/N_{\mathrm{sublat}}})$.

B. Generic sublattice coding algorithm

We start by discussing how we subdivide a lattice Λ into N_{sublat} sublattices. The basic requirement is that every symmetry group element operates only within either the sublattices or exchanges sublattices. We do not allow for symmetry elements that split up a sublattice into different sublattices. Therefore we make the following definition:

Definition (Sublattice stability). A decomposition,

$$\Lambda = \bigcup_{X=1}^{N_{\mathrm{sublat}}} \Lambda_X, \tag{19}$$

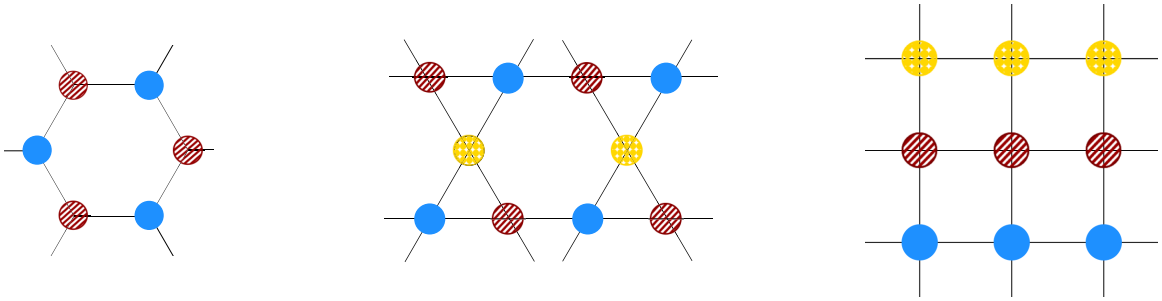

FIG. 2. Sublattice orderings for several common lattices. The sublattices are distinguished by different colors (shadings). Left: Two sublattices ordering in a honeycomb lattice. The sublattices are stable with respect to all spatial symmetries. Middle: Three sublattices ordering on a kagome lattice. The sublattices are stable with respect to all spatial symmetries. Right: Three sublattices ordering on a square lattice. The sublattices are stable with respect to all translational symmetries, horizontal and vertical reflections, and 180° rotations but not with respect to 90° rotations or diagonal reflections.

of a lattice Λ with symmetry group $\mathcal{G}$ into N_{sublat} disjoint sublattices Λ_X is called *sublattice stable* if every $g \in \mathcal{G}$ maps each Λ_X onto exactly one (possibly different) Λ_Y, i.e., for all $g \in \mathcal{G}$ and all Λ_X there exists a Λ_Y such that

$$g(\Lambda_X) = \Lambda_Y.$$

The set Λ_X is called the *X-sublattice* of Λ.

The notion of *sublattice stability* is illustrated in Fig. 2. The sublattices Λ_X are drawn in different colors (shadings). A translation by one unit cell keeps the sublattices of the honeycomb lattice invariant, whereas a 60° rotation exchanges the sublattices. For the kagome lattice in Fig. 2 a 60° rotation around a hexagon center for example cyclically permutes the three sublattices. One checks that for both the honeycomb and the kagome lattice in Fig. 2 all translational as well as all point group symmetries are sublattice stable, so different color (shading) sublattices are mapped onto each other. This is different for the square lattice in Fig. 2. Still here all translational symmetries just permute the sublattices, but a 90° rotation splits up a sublattice into different sublattices. Nevertheless, a 180° rotation keeps the sublattices stable, similarly a vertical or horizontal reflection. Therefore, only the reduced point group D2 instead of the full D4 point group for the square lattice fulfills the sublattice stability condition in this case. D2 and D4 denote the dihedral groups of order 4 and 8 with two- and fourfold rotations and reflections. Note that for a square lattice a two- or four-sublattice decomposition for which the full D4 point group is sublattice stable can be chosen instead. The choice of this particular sublattice decomposition just serves illustrational purposes.

From the definition of sublattice stability, it is clear that the total number of sites N has to be divisible by the number of sublattices N_{sublat}. The numbering of the lattice sites is chosen such that the lattice sites from $(X-1)N/N_{\text{sublat}}+1$ to XN/N_{sublat} belong to sublattice X. We choose the most significant bits in the integer representation to be the bits on sublattice 1. Similarly as in the previous section we define the following quantities.

Definition. For every sublattice Λ_X we define the following notions:

Sublattice symmetries:

$$\mathcal{G}_X \equiv \{g \in \mathcal{G} \mid g \text{ maps sublattice } X \text{ onto sublattice } 1\}. \quad (20)$$

Sublattice representative:

$$\texttt{Rep}_X(|\boldsymbol{\sigma}\rangle_X) \equiv h_X|\boldsymbol{\sigma}\rangle_X, \quad h_X = \underset{g\in\mathcal{G}_X}{\text{argmin}}\ \text{int}(g|\boldsymbol{\sigma}\rangle_X), \quad (21)$$

where $|\boldsymbol{\sigma}\rangle_X$ denotes the substate of $|\boldsymbol{\sigma}\rangle$ restricted on sublattice Λ_X.

Representative symmetries:

$$\texttt{Sym}_X(|\boldsymbol{\sigma}\rangle_X) \equiv \{g \in \mathcal{G}_X \mid g|\boldsymbol{\sigma}\rangle_X = \texttt{Rep}_X(|\boldsymbol{\sigma}\rangle_X)\}. \quad (22)$$

Sublattice symmetry action:

$$\texttt{SymmetryAction}_X(g, |\boldsymbol{\sigma}\rangle_X) = g|\boldsymbol{\sigma}\rangle_X. \quad (23)$$

The symmetries in $\mathcal{G}_X$ map the sublattice X onto the most significant bits. Therefore, the symmetry that minimizes the integer value in the orbit must be contained in the representative symmetries of a minimal sublattice representative:

$$g_{\boldsymbol{\sigma}} = \underset{g\in\mathcal{G}}{\text{argmin}}\, g|\boldsymbol{\sigma}\rangle \Rightarrow g_{\boldsymbol{\sigma}} \in \bigcup_{\substack{Y,\ \texttt{Rep}_Y(|\boldsymbol{\sigma}\rangle_Y)\\ \text{minimal}}} \texttt{Sym}_Y(|\boldsymbol{\sigma}\rangle_Y). \quad (24)$$

To find the minimizing symmetry $g_{\boldsymbol{\sigma}}$, we have to check only the symmetries yielding the minimal sublattice representative. The quantities $\texttt{Rep}_X(|\boldsymbol{\sigma}\rangle_X)$ and $\texttt{Sym}_X(|\boldsymbol{\sigma}\rangle_X)$ are stored in lookup tables, whose size scales as $O(2^{N/N_{\text{sublat}}})$. In order to quickly apply the symmetries, we can additionally store $\texttt{SymmetryAction}_X(g, |\boldsymbol{\sigma}\rangle_X)$ in another lookup table. The memory cost of doing so scales as $O(N_{\text{sym}}2^{N/N_{\text{sublat}}})$ and thus requires the most memory. The generic sublattice coding algorithm consists of two parts. The preparation of the lookup tables is shown as pseudocode in algorithm 1. The pseudocode of the actual algorithm for finding the representative using the lookup tables is shown in algorithm 2.

Algorithm 1. Preparation of lookup tables for sublattice coding algorithm

for each substate $|\boldsymbol{\sigma}_X\rangle$:
  for each sublattice X:
    compute $\texttt{Rep}_X(|\boldsymbol{\sigma}\rangle_X)$ Eq. (21), store it
    compute $\texttt{Sym}_X(|\boldsymbol{\sigma}\rangle_X)$ Eq. (22), store them
    for each symmetry $g \in \mathcal{G}$
      compute $\texttt{SymmetryAction}_X(g, |\boldsymbol{\sigma}\rangle_X)$, store it

Algorithm 2. Sublattice coding algorithm for finding the representative

Input: state $|\boldsymbol{\sigma}\rangle$
Output: representative $|\tilde{\boldsymbol{\sigma}}\rangle$ and g_{σ}
  Determine $\texttt{MinRep} = \min_X \{\texttt{Rep}_X(|\boldsymbol{\sigma}\rangle_X)\}$
  Set $|\tilde{\boldsymbol{\sigma}}\rangle = +\infty$
  for each sublattice Y with $\texttt{Rep}_Y(|\boldsymbol{\sigma}\rangle_Y) = \texttt{MinRep}$:
    for each symmetry $g \in \texttt{Sym}_Y(|\boldsymbol{\sigma}\rangle_Y)$:
      compute $g|\boldsymbol{\sigma}\rangle$ from $\texttt{SymmetryAction}_X(g, |\boldsymbol{\sigma}\rangle_X)$
      if $g|\boldsymbol{\sigma}\rangle < |\tilde{\boldsymbol{\sigma}}\rangle$:
        $|\tilde{\boldsymbol{\sigma}}\rangle \leftarrow g|\boldsymbol{\sigma}\rangle$
        $g_{\sigma} \leftarrow g$
  return $|\tilde{\boldsymbol{\sigma}}\rangle$, g_{σ}

Example. We consider the same state on a six-site chain lattice as in Fig. 1, but now using a three-sublattice decomposition in Fig. 3. We call the blue (solid) sublattice the A sublattice, the red (dashed) B, and the yellow (dotted) C. Notice, that due to different sublattice structure the labeling of the real space sites is different from the two sublattices case. In the three sublattices case, we are now given the state

$$|\boldsymbol{\sigma}\rangle = |\uparrow\uparrow\downarrow\downarrow\uparrow\downarrow\rangle. \tag{25}$$

Its substates are

$$\begin{aligned} |\boldsymbol{\sigma}\rangle_A &= |\uparrow\uparrow\rangle, \\ |\boldsymbol{\sigma}\rangle_B &= |\downarrow\downarrow\rangle, \\ |\boldsymbol{\sigma}\rangle_C &= |\uparrow\downarrow\rangle, \end{aligned} \tag{26}$$

with corresponding sublattice representatives

$$\begin{aligned} \texttt{Rep}_A(|\boldsymbol{\sigma}\rangle_A) &= |\uparrow\uparrow\rangle, \\ \texttt{Rep}_B(|\boldsymbol{\sigma}\rangle_B) &= |\downarrow\downarrow\rangle, \\ \texttt{Rep}_C(|\boldsymbol{\sigma}\rangle_C) &= |\downarrow\uparrow\rangle, \end{aligned} \tag{27}$$

and representative symmetries

$$\begin{aligned} \texttt{Sym}_A(|\boldsymbol{\sigma}\rangle_A) &= \{I, T_3\}, \\ \texttt{Sym}_B(|\boldsymbol{\sigma}\rangle_B) &= \{T_2, T_5\}, \\ \texttt{Sym}_C(|\boldsymbol{\sigma}\rangle_C) &= \{T_1\}. \end{aligned} \tag{28}$$

The minimal sublattice representative MinRep as in algorithm 2 is given by

$$\texttt{MinRep} = \texttt{Rep}_B(|\boldsymbol{\sigma}\rangle_B) = |\downarrow\downarrow\rangle. \tag{29}$$

The minimizing symmetry must now be in $\texttt{Sym}_B(|\boldsymbol{\sigma}\rangle_B) = \{T_2, T_5\}$. We see that

$$T_2|\boldsymbol{\sigma}\rangle = |\downarrow\downarrow\downarrow\uparrow\uparrow\uparrow\rangle < T_5|\boldsymbol{\sigma}\rangle = |\downarrow\downarrow\uparrow\downarrow\uparrow\uparrow\rangle. \tag{30}$$

Therefore, the representative $|\tilde{\boldsymbol{\sigma}}\rangle$ is given by

$$|\tilde{\boldsymbol{\sigma}}\rangle = |\downarrow\downarrow\downarrow\uparrow\uparrow\uparrow\rangle, \tag{31}$$

with the minimizing symmetry $g_{\sigma} = T_2$. Notice that this state differs from the one found in the two sublattices example since the labeling of the sites changes the integer representation of a state and thus the definition of the representative. Once a given labeling of sites is fixed, the representative is of course unique.

IV. DISTRIBUTED AND HYBRID MEMORY PARALLELIZATION

For reaching larger system sizes in ED computations a proper balance between memory requirements and computational costs has to be found. There are two major approaches when applying the Lanczos algorithm. The Hamiltonian matrix can either be stored in memory in some sparse-matrix format or generated on-the-fly every time a matrix-vector multiplication is performed. Storing the matrix is usually faster, yet memory requirements are higher. This approach is, for example, pursued by the software package SPINPACK [36]. A matrix-free implementation of the Lanczos algorithm usually needs more computational time since the matrix generation, especially in a symmetrized basis, can be expensive. Of course, the memory cost is drastically reduced since only a few vectors of the size of the Hilbert space have to be stored. It turns out that on current supercomputing infrastructures the main limitation in going to larger system sizes is indeed the memory requirements of the computation. It is thus often favorable to use a slower matrix-free implementation, as done by the software package $\mathcal{H}\Phi$ [37], for example. Due to these reasons, we also choose the matrix-free approach.

The most computational time in the Lanczos algorithm is used in the matrix-vector multiplication. The remaining types of operations are scalar multiplications, dot products of Lanczos vectors, or the diagonalization of the T-matrix, which are usually of negligible computational cost. Today's largest supercomputers are typically distributed memory machines, where every process has direct access only to a small part of the total memory. It is thus a nontrivial task to distribute data

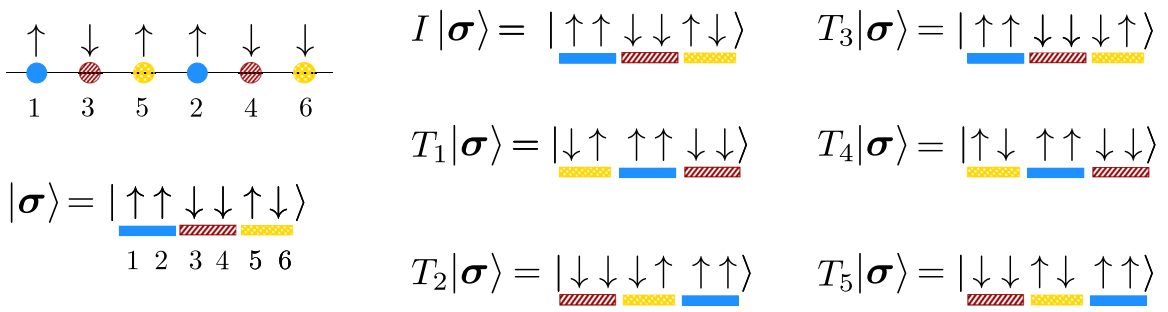

FIG. 3. Three sublattices coding of the spin state $|\boldsymbol{\sigma}\rangle$ on a six-site chain lattice and action of translation symmetries. The sites are enumerated such that sites 1 and 2 are on sublattice A (blue, solid), 3 and 4 on B (red, dashed), and 5 and 6 on C (yellow, dotted). The representative state with this enumeration of sites is given by $|\tilde{\boldsymbol{\sigma}}\rangle = T_2|\boldsymbol{\sigma}\rangle = |\downarrow\downarrow\downarrow\uparrow\uparrow\uparrow\rangle$

onto several processes and implement communication among them once remote memory has to be accessed. Also, when scaling the software to a larger amount of processes load balancing becomes important. The computational work should be evenly distributed among the individual processes in order to avoid waiting times in communication. In the following, we explain how we achieve this goal in our implementation using Message Passing Protocol (MPI).

Matrix-vector multiplication. The Hamiltonian can be written as a sum of nonbranching terms,

$$H = \sum_k H_k. \tag{32}$$

To perform the full matrix-vector multiplication we compute the matrix-vector multiplication for the nonbranching terms H_k and add up the results,

$$H|\psi\rangle = \sum_k H_k|\psi\rangle. \tag{33}$$

We denote by

$$\{|\boldsymbol{\sigma}_i\rangle\}, \quad i = 1, \ldots, D \tag{34}$$

a (possibly symmetry-adapted) basis of the Hilbert space. A wave function $|\psi\rangle$ is represented on the computer by storing its coefficients $\langle\boldsymbol{\sigma}_i|\psi\rangle$. Given an input vector,

$$|\psi_{\text{in}}\rangle = \sum_{i=1}^{D} \langle\boldsymbol{\sigma}_i|\psi_{\text{in}}\rangle|\boldsymbol{\sigma}_i\rangle, \tag{35}$$

we want to compute the coefficients $\langle\boldsymbol{\sigma}_i|\psi_{\text{out}}\rangle$ in

$$H_k|\psi_{\text{in}}\rangle = |\psi_{\text{out}}\rangle. \tag{36}$$

The resulting output vector $|\psi_{\text{out}}\rangle$ is given by

$$\begin{aligned} |\psi_{\text{out}}\rangle &= \sum_{i=1}^{D} \langle\boldsymbol{\sigma}_i|\psi_{\text{out}}\rangle|\boldsymbol{\sigma}_i\rangle = \sum_{i=1}^{D} \langle\boldsymbol{\sigma}_i|H_k|\psi_{\text{in}}\rangle|\boldsymbol{\sigma}_i\rangle \\ &= \sum_{i,j=1}^{D} c_k(\boldsymbol{\sigma}_j)\langle\boldsymbol{\sigma}_j|\psi_{\text{in}}\rangle\langle\boldsymbol{\sigma}_i|\boldsymbol{\sigma}'_j\rangle|\boldsymbol{\sigma}_i\rangle, \end{aligned} \tag{37}$$

where $c_k(\boldsymbol{\sigma}_j)$ and $|\boldsymbol{\sigma}'_j\rangle$ are given by

$$H_k|\boldsymbol{\sigma}_j\rangle = c_k(\boldsymbol{\sigma}_j)|\boldsymbol{\sigma}'_j\rangle. \tag{38}$$

Notice, that in a symmetry-adapted basis, evaluating $c_k(\boldsymbol{\sigma}_j)$ requires the evaluation of Eq. (6), where the sublattice coding technique can be applied. Clearly, we have

$$\langle\boldsymbol{\sigma}_i|\boldsymbol{\sigma}'_j\rangle = \begin{cases} 1 \text{ if } |\boldsymbol{\sigma}_i\rangle = |\boldsymbol{\sigma}'_j\rangle, \\ 0 \text{ else.} \end{cases} \tag{39}$$

For parallelizing the multiplication Eq. (37), we distribute the coefficients in the basis $\{|\boldsymbol{\sigma}_i\rangle\}$ onto the different MPI processes. This means we have a mapping,

$$\texttt{proc} : |\boldsymbol{\sigma}_i\rangle \to \{1, \ldots, n_{\text{procs}}\}, \tag{40}$$

that assigns to every basis state of the Hilbert space its MPI process number. Here n_{procs} denotes the number of MPI processes. In general, $|\boldsymbol{\sigma}_j\rangle$ and $|\boldsymbol{\sigma}'_j\rangle$ are not stored in the same process. Hence, the coefficient $c_k(\boldsymbol{\sigma}_j)\langle\boldsymbol{\sigma}_j|\psi_{\text{in}}\rangle$ has to be sent from the process number $\texttt{proc}(|\boldsymbol{\sigma}_j\rangle)$ to process number $\texttt{proc}(|\boldsymbol{\sigma}'_j\rangle)$. This makes communication between the processes necessary. This communication is buffered in our implementation, i.e., for every basis state $|\boldsymbol{\sigma}_j\rangle$ we first store the target basis state $|\boldsymbol{\sigma}'_j\rangle$ and the coefficient $c_k(\boldsymbol{\sigma}_j)\langle\boldsymbol{\sigma}_j|\psi_{\text{in}}\rangle$ locally. Once every local basis state has been evaluated, we perform the communication and exchange the information among all processes. This corresponds to an `MPI_Alltoallv` call in the MPI standard.

After this communication step, every process has to add the received coefficient to the locally stored coefficient $\langle\boldsymbol{\sigma}'_j|\psi_{\text{out}}\rangle$. For this, we have to search where the now locally stored coefficient of the basis state $|\boldsymbol{\sigma}'_j\rangle$ is located in memory. Typically, we keep a list of all locally stored basis states defining the position of the coefficients. This list is then searched for the entry $|\boldsymbol{\sigma}'_j\rangle$, which can also be time-consuming and needs to be done efficiently. We are thus facing the following challenges when distributing the basis states of the Hilbert space among the MPI processes:

(1) Every process has to know which process any basis state $|\boldsymbol{\sigma}_i\rangle$ belongs to.

(2) The storage of the information about the distribution should be memory efficient.

(3) The distribution of basis states has to be fair, in the sense that every process has a comparable workload in every matrix-vector multiplication.

(4) The search for a basis state within a process should be done efficiently.

We now propose a method to address these issues in a satisfactory way.

Distribution of basis states. The central point of our parallelization strategy is the proper choice of the distribution function $\texttt{proc}(\boldsymbol{\sigma})$ for the basis states in Eq. (40). We split up every basis state into prefix and postfix sites,

$$|\boldsymbol{\sigma}\rangle = |\underbrace{\sigma_1 \cdots \sigma_{n_{\text{prefix}}}}_{\text{prefix sites}} \quad \underbrace{\sigma_{n_{\text{prefix}}+1} \cdots \sigma_{n_{\text{prefix}}+n_{\text{postfix}}}}_{\text{postfix sites}}\rangle, \tag{41}$$

where n_{prefix} and n_{postfix} denote the number of prefix and postfix sites, respectively. We decide that states with the same prefix are stored in the same MPI process. The prefixes are randomly distributed among all the processes. We do this by using a hash function that maps the prefix bits onto a random but deterministic MPI process. This hash function can be chosen such that every process has a comparable amount of states stored locally. Moreover, a random distribution of states reduces load balance problems significantly since the communication structure is randomized. This is in stark contrast to distributing the basis states in a linear fashion. Thereby, single processes can often have a multiple of the workload than other processes, thus causing idle time in other processes.

By choosing this kind of random distribution of basis states, we also don't have to store any information about their distribution. This information is all encoded in the hash function. Nevertheless, we store the basis states belonging to a process locally in an array. Finding the index of a given basis state also requires some computational effort. Here we use the separation between prefix and postfix sites. We store the basis states in an ordered way. In this way, states belonging to the same prefix are aligned in memory as shown in Fig. 4. We can store the index of the first and the last states that

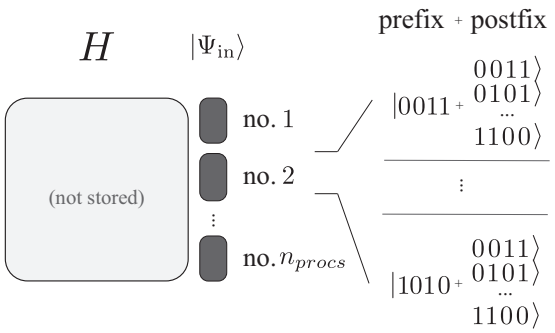

FIG. 4. Storage layout of the distributed Hilbert space. The prefixes are randomly distributed among the MPI processes using a hash function. States with same prefixes are mapped to the same process. Within a process, the states are ordered lexicographically. The Hamiltonian matrix is not stored.

belong to a given prefix. To find the index of a given state we can now look up the first and last index of the prefix of this state and perform a binary search for the state between these two indices. This reduces the length of the array we have to perform the binary search on and, hence, reduces the computational effort in finding the index. For implementing this procedure we need two data structures locally stored on each process:

(1) An array $\texttt{Basis}(i)$ storing all the basis states,

$$\texttt{Basis}(i) = |\boldsymbol{\sigma}_i\rangle, \quad i = 1, \ldots, D. \tag{42}$$

(2) An associative array $\texttt{Limits}(|\boldsymbol{\sigma}_{\text{prefix}}\rangle)$ storing the map

$$\texttt{Limits}(|\boldsymbol{\sigma}_{\text{prefix}}\rangle) = [\texttt{begin}(|\boldsymbol{\sigma}_{\text{prefix}}\rangle), \texttt{end}(|\boldsymbol{\sigma}_{\text{prefix}}\rangle)] \tag{43}$$

where $\texttt{begin}(|\boldsymbol{\sigma}_{\text{prefix}}\rangle)$ denotes the index of the first state with prefix $|\boldsymbol{\sigma}_{\text{prefix}}\rangle$ and $\texttt{end}(|\boldsymbol{\sigma}_{\text{prefix}}\rangle)$ denotes the index of the last state with this prefix in the array $\texttt{Basis}(i)$, $|\boldsymbol{\sigma}_{\text{prefix}}\rangle = |\sigma_1 \cdots \sigma_{n_{\text{prefix}}}\rangle$.

In algorithm 3 we summarize how to prepare these data structures. The parallel matrix-vector multiplication in pseudocode is shown in algorithm 4. When working in the symmetry-adapted basis, the lookup tables of the sublattice coding method need to be accessible to every MPI process. One way to achieve this is, of course, that every process generates its own lookup tables. However, in present-day supercomputers, several processes will be assigned to the same physical machine sharing the same physical memory. To save memory, the lookup tables are stored only once on a computing node. Its processes can then access the lookup tables via shared memory access. In our code, we use POSIX shared memory functions [38] to implement this hybrid parallelization.

Algorithm 3. Preparation of the distributed and symmetrized Hilbert space

Perform the following steps on every process in parallel (no communication necessary)
`myid` denotes the number of the current MPI process
prepares data structures `Basis`, `Limits` on each process
for each prefix spin configuration $|\boldsymbol{\sigma}_{\text{prefix}}\rangle = |\sigma_1 \cdots \sigma_{n_{\text{prefix}}}\rangle$:
  if $\texttt{proc}(|\boldsymbol{\sigma}_{\text{prefix}}\rangle) \neq \texttt{myid}$:
    continue
  else:
    `begin` = length(`Basis`)
    for each spin configuration $|\boldsymbol{\sigma}\rangle$ with prefix $|\boldsymbol{\sigma}_{\text{prefix}}\rangle$:
      compute representative $|\tilde{\boldsymbol{\sigma}}\rangle$ of $|\boldsymbol{\sigma}\rangle$
      if $|\boldsymbol{\sigma}\rangle = |\tilde{\boldsymbol{\sigma}}\rangle$:
        append $|\boldsymbol{\sigma}\rangle$ to `Basis`
    `end` = **length**(`Basis`)
    if end ≠ begin
      insert ($|\boldsymbol{\sigma}_{\text{prefix}}\rangle$, begin, end) to `Limits`

Algorithm 4. Parallel matrix-vector multiply for a nonbranching term H_k

Input: input wave function $|\psi_{\text{in}}\rangle$
Output: matrix-vector product $|\psi_{\text{out}}\rangle = H_k|\psi_{\text{in}}\rangle$
▷ Preparation and sending step (communication may be buffered)
for each basis state $|\boldsymbol{\sigma}_j\rangle$ stored locally in `Basis`
  · apply nonbranching H_k and use sublattice coding technique to compute $c_k(\boldsymbol{\sigma}_j)$ and $|\boldsymbol{\sigma}'_j\rangle$,
  $$H_k|\boldsymbol{\sigma}_j\rangle = c_k(\boldsymbol{\sigma}_j)|\boldsymbol{\sigma}'_j\rangle.$$
  · compute $c = c_k(\boldsymbol{\sigma}_j)\langle\boldsymbol{\sigma}_j|\psi_{\text{in}}\rangle$
  · send the pair $(|\boldsymbol{\sigma}'_j\rangle, c)$ to process no. $\texttt{proc}(|\boldsymbol{\sigma}'_j\rangle)$
▷ Receiving and search step
for each pair $(|\boldsymbol{\sigma}'_j\rangle, c)$ received
  · determine indices (`begin`, `end`) from $\texttt{Limits}(|\boldsymbol{\sigma}'_j\rangle)$
  · determine index i of $|\boldsymbol{\sigma}'\rangle$ by binary search in array `Basis` between (`begin`,`end`)
  · Set $\langle\boldsymbol{\sigma}'_j|\psi_{\text{out}}\rangle[i] \leftarrow c$

V. BENCHMARKS

In order to assess the power of the methods proposed in the previous sections, we performed test runs to compute ground state energies. We considered the Heisenberg antiferromagnetic spin-1/2 nearest-neighbor model,

$$H = J \sum_{\langle i,j \rangle} \boldsymbol{S}_i \cdot \boldsymbol{S}_j, \quad J = 1, \tag{44}$$

on four different lattice geometries: square (48 sites), triangular (48 sites), kagome (48 sites), and square (50 sites). Figure 5 shows the simulation clusters and the sublattice structure we used. The benchmarks were performed on three different supercomputers. The Vienna Scientific Cluster VSC3 is built up from more than 2020 nodes with two Intel Xeon E5-2650v2, 2.6 GHz, eight-core processors, the supercomputer Hydra at the Max Planck Supercomputing and Data Facility in Garching, Germany, with more than 3500 nodes with 20-core Intel Ivy Bridge 2.8 GHz processors, and the System B Sekirei at the Institute for Solid State Physics of the University of Tokyo with more than 1584 nodes with two Intel Xeon E5-2680v3 12-core 2.5 GHz processors. Both the Hydra and Sekirei use InfiniBand FDR interconnect, whereas the VSC3 uses Intel TrueScale Infiniband for network communication.

The benchmarks are summarized in Table I. We make use of all translational, certain point group symmetries and spin-flip symmetry. We show the memory occupied by a single lookup table for the symmetries. Since we use a single

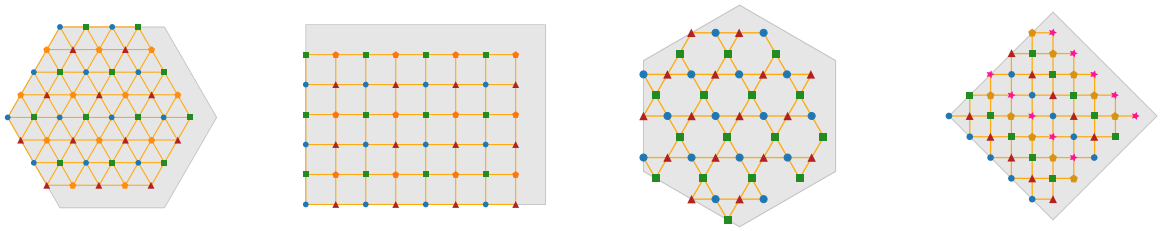

FIG. 5. Geometries of Heisenberg spin-1/2 model benchmarks. Different colors (symbols) show the sublattice structure used for the sublattice coding technique. The gray background shows the Wigner-Seitz cell defining the periodicity of the lattice. Left: Triangular lattice, 48 sites, four sublattices structure. Middle left: Square lattice, 48 sites, four sublattices structure. Middle right: kagome lattice, 48 sites, three sublattices structure. Right: Square lattice, 50 sites, five sublattices structure.

buffered and blocking all-to-all communication in the implementation, it is straightforward to measure the percentage of time spent for MPI communication by taking the time before the communication call and afterward. In order to validate the results of our computation, we compared the results of the unfrustrated square case to quantum Monte Carlo computations of the ground state energy. We used a continuous time world-line Monte Carlo code [39] with 10^5 thermalization and 10^6 measurements at temperature $T = 0.01$. The computed energies per site are $E/N = -0.676013 \pm 2 \times 10^{-5}$ for the 48-site square cluster and $E/N = -0.67512 \pm 2 \times 10^{-5}$ for the 50-site cluster. The actual values computed with ED are within the error bars. The ground state energy of the kagome Heisenberg antiferromagnet on 48 sites has been previously computed [14] with a specialized code and agrees with our results. We see that the amount of time spent for communication is different for the three supercomputers. On Sekirei, a parallel efficiency of 61% on 3456 cores has been achieved.

Results for running the same problem on various numbers of processors are shown in Fig. 6. We chose two different problems for two sets of number of cores, the Heisenberg antiferromagnet with additional next-nearest neighbor and third nearest-neighbor interactions on a 40-site square lattice and the Heisenberg antiferromagnet on a 48-site triangular lattice. We observe almost ideal scaling behavior up to 4096 MPI processes. Hence, the parallelization strategy described above successfully solves load balancing problems in the MPI communication. This benchmark has been performed on the Curie supercomputer at GENCI-TGCC-CEA, France.

VI. DISCUSSION

The sublattice coding technique presented in Sec. II allows for fast and memory-efficient evaluation of the matrix elements in a symmetry-adapted basis Eq. (6). Still, the construction of the sublattices imposes restrictions on the geometry of the simulation cluster. A sublattice construction with N_{sublat} sublattices at least requires the number of sites to be divisible by N_{sublat}. The sublattice coding technique yields no advantages for lattice samples that have a prime number of sites. For lattices with several basis sites per unit cell a natural sublattice decomposition exists. The sublattices are given by the lattices defined by the corresponding basis sites. This is the case for the honeycomb and kagome lattice, where a natural two- (resp. three-) sublattice decomposition exists; cf. Fig. 2.

The sublattice decomposition for a given lattice is not unique. This can be seen in the case of the 50-site square lattice, whose five-sublattice decomposition is shown in Fig. 5. As a bipartite square lattice, it also allows for a two-sublattice decomposition. Three- and four-sublattice decompositions exist for other square lattice clusters as well (see, e.g., Fig. 5). Hence, for a given simulation cluster there may exist

TABLE I. Benchmark results for various problems on the three discussed different supercomputer systems. The employed symmetries include translational, point group, and spinflip symmetry. We show the total memory used by all MPI processes and the memory used by the lookup tables for the sublattice-coding technique. We also show the amount of time spent for communication. For labeling the ground state representations, Γ denotes the (0,0) and M the (π, π) point in the Brillouin zone. A1 denotes the trivial point group representation, and even or odd denotes the spinflip symmetry representation. The energy is given in units of the nearest neighbor coupling constant $J = 1$.

Geometry computer	Triangular 48 Sekirei	Square 48 VSC3	Kagome 48 Hydra	Square 50 Sekirei
Point group	D6	D2	D6	D2
No. of symmetries	1152	384	384	400
Dimension	2.8×10^{10}	8.3×10^{10}	8.4×10^{10}	3.2×10^{11}
No. of cores	3456	8192	10 240	3456
Total memory	2.5 Tb	N.A.	N.A.	15.5 Tb
Memory lookup	151 Mb	50 Mb	604 Mb	17 Mb
Time/matrix-vector multiplication	399 s	1241 s	258 s	3304 s
% Comm. time	39%	77%	48%	39%
Ground state sector	Γ.A1.even	Γ.A1.even	Γ.A1.even	M.A1.odd
Ground state energy	−26.812 945 271 5	−32.447 359 872 8	−21.057 787 063 5	−33.755 101 931 5

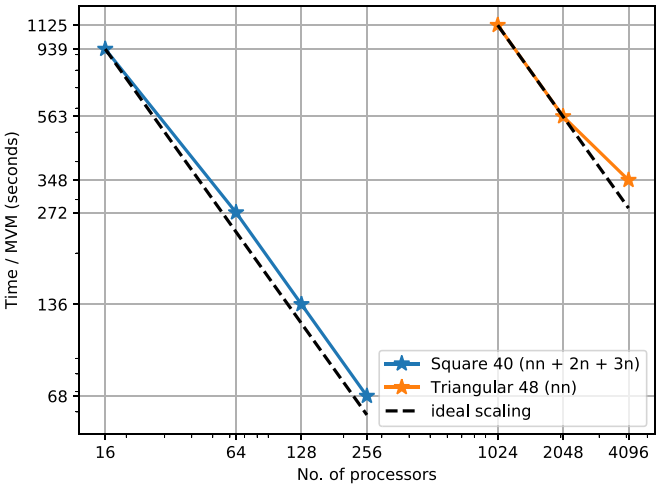

FIG. 6. Scaling behavior of the parallelization. The time for one matrix-vector multiplication in seconds is compared for the total number of processes. Both axes are scaled logarithmically. The matrix is the Hamiltonian of the Heisenberg spin 1/2 antiferromagnet on a 40-site cluster with additional next-nearest neighbor and third-nearest neighbor interactions for the benchmark on 16, 64, 128, and 256 cores. For 1024, 2048, and 4096 cores the matrix is the Hamiltonian of the Heisenberg spin 1/2 antiferromagnet on a 48-site triangular cluster. The Hamiltonian is considered in its ground state sector. We observe almost ideal scaling behavior up to 4096 processors.

more than one sublattice decomposition. Most of the high-symmetry square and triangular lattice samples possess at least one-sublattice decomposition in two or more sublattices. Still, a given sublattice decomposition may restrict the symmetry group if certain symmetry elements split up sublattices. This is, for example, the case for the 50-site square lattice in Fig. 5. While the cluster itself has a full fourfold rotational and reflectional symmetry, the 90° rotation is not sublattice stable. Therefore, only the 180° rotation and reflection symmetry have been used as point group symmetries in the computation.

Increasing the number of sublattices decreases the memory required for storing the lookup tables. The computational effort for computing the representative as in algorithm 2 increases linearly in the number of sublattices. Also, smaller sublattices yield more potential representative symmetries [Eq. (22)] that have to be applied to the spin configuration. In principle, there is no restriction on the number of sublattices, and the proper choice depends on the geometry of the simulation cluster, the available memory, and the desired speed. Fewer sublattices allow algorithm 2 to evaluate matrix elements faster.

The method of distributing the basis states of the Hilbert space is independent of the sublattice coding algorithm. Hence, this kind of parallelization can also be applied to problems without symmetries, like disordered systems. All information about the distribution of basis states is encoded by the hash function, which can be of a rather simple type to achieve a balanced distribution.

One main motivation for performing large-scale ED computations is reaching system sizes for which high-symmetry clusters are available. The possibility to simulate 48 spin-1/2 particles gives access to the interesting triangular and kagome lattices shown in Fig. 5. These samples both have full sixfold rotational and reflectional symmetry. In reciprocal space, these clusters accommodate both the K point and even the M point for the triangular case. This feature is important in order to distinguish different phases with different ordering vectors. For the square lattice case, an interesting 52-site cluster with fourfold rotational symmetry exists featuring the (π, π) and $(\pi, 0)$ point in reciprocal space. A study of the Heisenberg model with next-nearest neighbor interactions on this cluster can, therefore, yield valuable insights into nature of the intermediate phase, whose nature is not fully understood as of today. The methods proposed in this paper allow for these calculations on large present-day supercomputers, since the Hilbert space dimension of this problem is roughly four times larger than the investigated 50-site case and a four-sublattice decomposition is available.

Apart from spin systems, the sublattice coding algorithm also applies to fermionic systems, when the Hamiltonian is expressed in the occupation number basis. The occupation numbers of the orbitals on the respective sublattices define the sublattice configurations. In addition to computing a representative and representative symmetry, a Fermi sign has to be computed to evaluate matrix elements, which can be done efficiently.

VII. CONCLUSION

We proposed the generic sublattice coding algorithm for making efficient use of discrete symmetries in large-scale ED computations. The method can be used flexibly on most lattice geometries and requires only a reasonable amount of memory for storing the lookup tables. The parallelization strategy for distributed memory architectures we discussed includes a random distribution of the Hilbert space among the parallel processes. Lookup tables of the sublattice coding technique are stored only once per node and are accessed via shared memory. Using these techniques, we showed that computations of spin-1/2 models of up to 50 spins have now become feasible.

ACKNOWLEDGMENTS

The ED calculations of the 50-site cluster have been performed using the facilities of the Supercomputer Center, Institute for Solid State Physics, University of Tokyo. The scaling benchmarks in Fig. 6 have been performed on the supercomputer Curie (GENCI-TGCC-CEA, France). We especially thank Synge Todo, Roderich Moessner, and Sylvain Capponi for making some of these simulations possible. A.W. acknowledges support through the Austrian Science Fund project I-1310-N27 (DFG FOR1807) and the Marietta Blau-Stipendium of OeAD-GmbH, financed by the Austrian Bundesministeriums für Wissenschaft, Forschung und Wirtschaft (BMWFW). Further computations for this paper have been carried out on VSC3 of the Vienna Scientific Cluster, the supercomputer Hydra at the Max Planck Supercomputing and Data Facility in Garching, Germany.

[1] Z. Gan and R. J. Harrison, in *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing, SC '05*, Seattle, WA, USA (IEEE, 2005), pp. 22–22.

[2] P. Sternberg, E. G. Ng, C. Yang, P. Maris, J. P. Vary, M. Sosonkina, and H. V. Le, in *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, SC '08* (IEEE Press, Piscataway, NJ, 2008), pp. 15:1–15:12.

[3] J. P. Vary, P. Maris, E. Ng, C. Yang, and M. Sosonkina, J. Phys. Conf. Ser. **180**, 012083 (2009).

[4] P. Maris, M. Sosonkina, J. P. Vary, E. Ng, and C. Yang, Procedia Comput. Sci. **1**, 97 (2010).

[5] S. Rychkov and L. G. Vitale, Phys. Rev. D **91**, 085011 (2015).

[6] J. Oitmaa and D. D. Betts, Can. J. Phys. **56**, 897 (1978).

[7] A. Wietek and A. M. Läuchli, Phys. Rev. B **95**, 035141 (2017).

[8] A. Wietek, A. Sterdyniak, and A. M. Läuchli, Phys. Rev. B **92**, 125122 (2015).

[9] E. Dagotto and A. Moreo, Phys. Rev. B **39**, 4744 (1989).

[10] P. W. Leung and V. Elser, Phys. Rev. B **47**, 5459 (1993).

[11] H. J. Schulz, T. A. L. Ziman, and D. Poilblanc, J. Phys. (France) **6**, 675 (1996).

[12] J. Richter and J. Schulenburg, Eur. Phys. J. B **73**, 117 (2010).

[13] A. M. Läuchli, R. Johanni, and R. Moessner, An exact diagonalization perspective on the $s = 1/2$ kagome Heisenberg antiferromagnet, talk at the 2012 KITP Fragnets Program, http://online.kitp.ucsb.edu/online/fragnets12/laeuchli/.

[14] A. M. Läuchli, J. Sudan, and R. Moessner, arXiv:1611.06990.

[15] H. Q. Lin and J. E. Hirsch, Phys. Rev. B **35**, 3359 (1987).

[16] Y. Hasegawa and D. Poilblanc, Phys. Rev. B **40**, 9035 (1989).

[17] J. Bonča, P. Prelovšek, and I. Sega, Phys. Rev. B **39**, 7074 (1989).

[18] D. Poilblanc, Phys. Rev. B **52**, 9201 (1995).

[19] S. He, S. H. Simon, and B. I. Halperin, Phys. Rev. B **50**, 1823 (1994).

[20] R. H. Morf, N. d'Ambrumenil, and S. Das Sarma, Phys. Rev. B **66**, 075408 (2002).

[21] T. Neupert, L. Santos, C. Chamon, and C. Mudry, Phys. Rev. Lett. **106**, 236804 (2011).

[22] A. M. Läuchli, Z. Liu, E. J. Bergholtz, and R. Moessner, Phys. Rev. Lett. **111**, 126802 (2013).

[23] M. Schuler, S. Whitsitt, L.-P. Henry, S. Sachdev, and A. M. Läuchli, Phys. Rev. Lett. **117**, 210401 (2016).

[24] S. Whitsitt, M. Schuler, L.-P. Henry, A. M. Läuchli, and S. Sachdev, Phys. Rev. B **96**, 035142 (2017).

[25] H. Q. Lin, Phys. Rev. B **42**, 6561 (1990).

[26] A. Weiße, Phys. Rev. E **87**, 043305 (2013).

[27] R. P. Feynman, Int. J. Theor. Phys. **21**, 467 (1982).

[28] H. Häffner, W. Hänsel, C. F. Roos, J. Benhelm, D. Chek-al kar, M. Chwalla, T. Körber, U. D. Rapol, M. Riebe, P. O. Schmidt, C. Becher, O. Gühne, W. Dür, and R. Blatt, Nature (London) **438**, 643 (2005).

[29] D. Castelvecchi, Nature (London) **541**, 9 (2017).

[30] T. Häner and D. S. Steiger, in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '17* (ACM, New York, 2017), pp. 33:1–33:10.

[31] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, T. Magerlein, E. Solomonik, and R. Wisnieff, arXiv:1710.05867 [quant-ph].

[32] A. Gheorghiu, T. Kapourniotis, and E. Kashefi, Theory Comput. Syst. (2018), doi: 10.1007/s00224-018-9872-3.

[33] C. Lanczos, J. Res. Natl. Bur. Stand. **45**, 255 (1950).

[34] L. D. Landau and E. M. Lifschitz, *Lehrbuch der Theoretischen Physik, Vol. III, Quantenmechanik*, 8th ed. (Akademie-Verlag, Berlin, 1988).

[35] A. M. Läuchli, *Introduction to Frustrated Magnetism: Materials, Experiments, Theory* (Springer, Berlin, 2011), pp. 481–511.

[36] J. Schulenburg, SPINPACK software (2017), https://www-e.uni-magdeburg.de/jschulen/spin/.

[37] M. Kawamura, K. Yoshimi, T. Misawa, Y. Yamaji, S. Todo, and N. Kawashima, Comput. Phys. Commun. **217**, 180 (2017).

[38] SHM_OVERVIEW(7), Linux Programmer's Manual, 4th ed. (2017), http://man7.org/linux/man-pages/man7/shm_overview.7.html.

[39] S. Todo and K. Kato, Phys. Rev. Lett. **87**, 047203 (2001).