The Journal of Chemical Physics

RESEARCH ARTICLE | SEPTEMBER 04 2024

Markov-chain sampling for long-range systems without evaluating the energy

Special Collection: Monte Carlo methods, 70 years after Metropolis et al. (1953)

Gabriele Tartero ; Werner Krauth

J. Chem. Phys. 161, 094106 (2024)
https://doi.org/10.1063/5.0225561

AIP Publishing

The Journal of Chemical Physics ARTICLE pubs.aip.org/aip/jcp

Markov-chain sampling for long-range systems without evaluating the energy

Cite as: J. Chem. Phys. 161, 094106 (2024); doi: 10.1063/5.0225561
Submitted: 25 June 2024 • Accepted: 13 August 2024 •
Published Online: 4 September 2024

Gabriele Tartero[1,a)] and Werner Krauth[1,2,3,b)]

AFFILIATIONS

[1] Laboratoire de Physique de l'École Normale Supérieure, ENS, Université PSL, CNRS, Sorbonne Université, Université Paris Cité, Paris, France
[2] Rudolf Peierls Centre for Theoretical Physics, Clarendon Laboratory, Oxford OX1 3PU, United Kingdom
[3] Simons Center for Computational Physical Chemistry, New York University, New York, New York 10012, USA

Note: This paper is part of the JCP Special Topic on Monte Carlo methods, 70 years after Metropolis *et al.* (1953).

[a)] gabriele.tartero@phys.ens.fr
[b)] **Author to whom correspondence should be addressed:** werner.krauth@ens.fr

ABSTRACT

In past decades, enormous effort has been expended to develop algorithms and even to construct special-purpose computers in order to efficiently evaluate total energies and forces for long-range-interacting particle systems, with the particle-mesh Ewald and the fast multipole methods as well as the "Anton" series of supercomputers serving as examples for biomolecular simulations. Cutoffs in the range of the interaction have also been used for large systems. All these methods require extrapolations. Within Markov-chain Monte Carlo, in thermal equilibrium, the Boltzmann distribution can, however, be sampled natively without evaluating the total energy. Using as an example the Lennard-Jones interaction, we review past attempts in this direction and then discuss in detail the class of cell-veto algorithms that allow for the fast, native sampling of the Boltzmann distribution without any approximation, extrapolation, or cutoff even for the slowly decaying Coulomb interaction. The computing effort per move remains constant with increasing system size, as we show explicitly. We provide worked-out illustrations and pseudocode representations of the discussed algorithms. Python scripts are made available in an associated open-source software repository.

I. INTRODUCTION

One of the aims of computational science is to analyze, and often solve, model systems in fields from subatomic particles to condensed-matter physics and chemistry, to galaxies, and to the universe. The power of modern computers appears without limits when compared to the early electronic devices from only two generations ago. Algorithms have also much evolved, as has the ease with which computers are interfaced and their output is processed. Nevertheless, certain limits persist. The first, broadly speaking, is a limitation in space: one tries to simulate large, possibly infinite systems but is restricted by the finite extent of computer memories. Second, one may aim for long-time simulations, but the necessary evolution equations may not allow one to reach the relevant temporal scales. The third limitation is in the type of couplings between particles or fields. They may be of many-body nature, including quantum and long-range potentials, and may even at present require simplifications.[1]

In classical condensed-matter particle systems in thermal equilibrium, the limitations on space, time, and what could be called "range" can again be illustrated. In contexts from statistical mechanics to molecular simulation in biochemistry, one may want to analyze large, almost infinite systems, but only in exceptional cases is it possible to simulate them directly.[2–6] Prominent coarse-graining strategies were developed[7] in order to reach larger and larger sizes. On the other hand, it is the change of behavior of finite systems with size, the famous finite-size scaling, which often provides crucial information on a physical model.[8] One also may want to reach essentially infinite simulation times, as the equilibrium state is approached from an initial configuration only in this limit.

Computer computations are by definition of finite duration and, within Markov-chain Monte Carlo, it is only in exceptional cases possible to reach the infinite-time limit directly.[9,10]

The approximation of complex interaction potentials, the above "range" limitation, is what we are concerned with in this paper, again in the framework of equilibrium physics. We consider systems of particles at a given temperature and governed by the Boltzmann distribution. Such systems can be simulated by molecular dynamics or sampled by the Monte Carlo method. Particle systems with Coulomb or Lennard-Jones interactions will serve as examples. The Coulomb interaction decays as $1/r$ with the distance r between charged particles. Its long-range nature derives from the masslessness of the photon. The Lennard-Jones potential describes the interaction between uncharged, non-polar atoms. Repulsive at small distances, its attractive $1/r^6$ behavior at large distances describes the London dispersive force.[11] For both the Coulomb and the Lennard-Jones potentials, the long-range nature is thus rooted in profound physical principles, and there is a strong incentive to maintain it in the modeling. However, long-range potentials pose severe problems for computation, which arise naïvely because the total energy of a system of N particles interacting in pairs consists of $N(N-1)/2$ terms. Moving one particle changes $N-1$ terms in the total energy. Likewise, the force on a given particle is composed of $N-1$ terms.

Computational science, in the past decades, has focused on how to evaluate total energies for long-range interaction potentials or, similarly, how to evaluate the forces. For the case of the Coulomb interaction, the potential or the forces are computed by interpolating charges to a grid, then solving Poisson's equation in discretized momentum space using a fast Fourier transform. The "Anton" series of supercomputers has been designed in order to optimize interpolation and force evaluation,[12] yielding spectacular speedups.[13] The discretization error of these so-called particle-mesh Ewald methods disappears only for vanishing grid size. The fast multipole method[14] presents another extrapolation of the total energy in terms of moments of the charge distribution. Very often also, the Coulomb interaction is cut off beyond a certain radius, although this profoundly modifies the underlying physics of the models. Discretization and cutoffs both call for extrapolations. For decades, the Lennard-Jones potential was cut off at a finite distance, although artifacts introduced by the cutoff on phase boundaries[15,16] and on interface effects[17,18] were pointed out repeatedly. In recent years, this problem has been identified, and the particle-mesh Ewald approach has extended to the Lennard-Jones system.[19] However, this creates an avalanche of problems, as the empirical interaction potentials used for the sampling are themselves fitted (originally with a cutoff) and then have to be reparametrized.[20]

A Monte Carlo algorithm typically consists in a sequence of elementary proposed moves (sampled from a given probability distribution) that are either accepted or rejected, typically by evaluating the incurred change of the total energy.[21] There is much liberty in which moves to propose and how to accept or reject them with a "filter." The proposed moves are simple, and they can be the same for different problems. For example, they can be random displacements along the positive and negative x, y, and z directions. The decision to accept or reject a rather simple proposed move makes the Markov chain converge to the Boltzmann distribution and, for this reason, Markov-chain Monte Carlo is a decision problem at heart. The only condition to be satisfied is that the Boltzmann distribution be stationary with respect to the moves encoded in a transition matrix.[22] This differs from molecular dynamics, whose moves specifically implement discrete-time Hamiltonian (gradient) dynamics.

In this paper, we discuss methods for sampling positions $\mathbf{x}$ in a given sample space that are distributed according to the Boltzmann distribution $\pi(\mathbf{x}) = \exp[-\beta U(\mathbf{x})]$ at inverse temperature β without evaluating the total energy $U(\mathbf{x})$. This approach differs from the previously mentioned strategies for evaluating the total energy or the forces through interpolation or through the multipole expansion. It leads to the fast *native* sampling algorithms for many-body systems interacting through long-range potentials (the word "native" refers to methods that are automatically correct, in our case that are free of cutoffs, discretizations, or time-stepping errors). In the main part of the paper, we discuss in detail the cell-veto class of algorithms,[23] which rely on two key ingredients: First, the accept/reject decision that, as discussed, defines the Monte Carlo method is split, using the factorized Metropolis filter,[24] into a large number of independent factor decisions. These decisions are connected through a "consensus principle" that we will discuss in detail rather than through the usual changes in total energy. Second, the accept/reject decision for a factor is generally arrived at in two steps using two "pebbles"[25] that correspond to the "thinning" of an inhomogeneous Poisson process:[26] first, an approximate (conservative) decision provisionally rejects "too many moves," and, second, a correction step accepts some of the provisional rejects so that the overall rejection probability comes out to be correct. We provide a step-by-step introduction to this class of algorithms, together with heuristics, pseudocodes, and simplified Python scripts that are made available in an open-source software repository (see the Appendix for access information and for a correspondence between the main text, our pseudocodes, and the Python scripts). We thus hope to facilitate access to a class of algorithms that have remained cryptic and have never been presented in the context of decision problems and their possible extensions.[27–30]

II. FROM LONG-RANGE POTENTIALS TO THE CELL-VETO ALGORITHM

We concentrate on long-range pair potentials, leaving aside many-body terms, although they can be handled as well.[30,31] Several strategies for treating the long-range nature are discussed in Sec. II A, such as solving the accept/reject decision with approximate potentials[27] and factorizing the total energy into short-range and long-range contributions.[28] The factorized Metropolis filter generalizes this approach (Sec. II B). Together with the concept of bounding potential, it lies at the heart of the native $\mathcal{O}(1)$ (per move) cell-veto algorithms, of which we present a naïve (that is, correct but inefficient) version in Sec. II C before sharpening it in Sec. III.

A. Potentials, Metropolis algorithm

Specifically, we consider throughout this paper N particles inside a two-dimensional periodic square box of size L interacting via the Lennard-Jones pair potential,[32]

$$U^{\mathrm{LJ}}(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]. \tag{1}$$

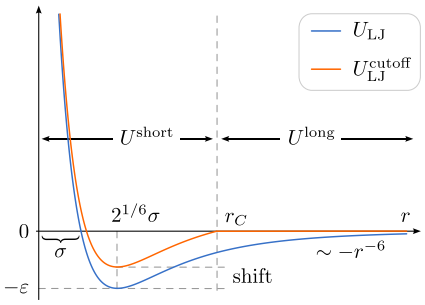

FIG. 1. Lennard-Jones potential with short-range and long-range parts separated by a cutoff r_c and a shifted cutoff variant that vanishes beyond r_c. For large r, $U^{\rm LJ}(r) \sim r^{-6}$ is attractive.

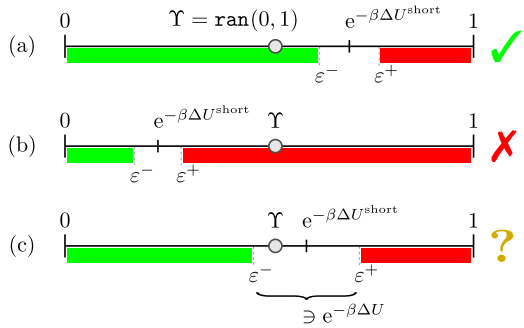

FIG. 2. Decisions in MCMC. In the Metropolis filter of Eq. (4), the "accept" decision in case (a) and the "reject" decision in case (b) can be based on a short-range approximation $e^{-\beta\Delta U^{\rm short}}$ of $e^{-\beta\Delta U}$, as the random number Υ lies outside the confidence window of Eq. (5). In case (c), r_c must be increased.

At large distances, its attractive tail vanishes as $1/r^6$ (see Fig. 1). Our Python scripts in the Appendix are for this two-dimensional system, with N particles interacting with the potential of Eq. (1), but they generalize to higher dimensions and to real-life interactions mixing short-range and long-range terms as well as pairwise and few-body contributions (see Ref. 30 for real-life situations handled with the JeLLyFysh software[33,34]). For the most part, a cutoff means that one can separate the interaction potential $U(r)$ into a short-range and a long-range contribution,

$$U(r) = \underbrace{U(r)\Theta(r_c - r)}_{U^{\rm short}} + \underbrace{U(r)\Theta(r - r_c)}_{U^{\rm long}}, \tag{2}$$

where $\Theta(z)$ is the unit-ramp function, which is zero for negative z and one for positive z. The Boltzmann weight of a particle configuration $\mathbf{x}$ with pairs interacting with the potential U of Eq. (2) then becomes the product of Boltzmann weights of the constituents,

$$\pi(\mathbf{X}) = \pi^{\rm short}(\mathbf{X})\pi^{\rm long}(\mathbf{X}). \tag{3}$$

In the following, we refer to the total energy of a configuration $\mathbf{x}$ as $U(\mathbf{x})$, with the Boltzmann weights $\pi(\mathbf{x}) = \exp\left[-\beta U(\mathbf{x})\right]$ and $\pi^{\rm short}(\mathbf{x}) = \exp\left[-\beta U^{\rm short}(\mathbf{x})\right]$, etc., and to the pair potential of two particles at a distance r as $U(r)$.

Commonly, a cutoff replaces $U(r)$ by its short-range part, and the long-range part is set to zero (see Fig. 1, for the Lennard-Jones potential). In that case, it is customary to shift $U^{\rm short}$ so that the approximate potential is continuous. More elaborate procedures are common in order to render the approximate potential better behaved around r_c, as is often required for molecular dynamics.

In the Metropolis algorithm, a random particle at position $\mathbf{x} \in \mathbf{X}$ is often selected for random displacement to $\mathbf{x}' = \mathbf{x} + \Delta\mathbf{x}$. The move from $\mathbf{X}$ to $\mathbf{X}'$ (in the latter, $\mathbf{x}$ is replaced by $\mathbf{x}'$) is accepted with a probability given by the Metropolis filter

$$\mathcal{P}^{\rm Met}(\mathbf{X} \to \mathbf{X}') = \min\left[1, \exp\left(-\beta\Delta U\right)\right], \tag{4}$$

where $\Delta U = U(\mathbf{x}') - U(\mathbf{x})$. See the Appendix for a basic Python script implementing the Metropolis algorithm for the two-dimensional Lennard-Jones system without a cutoff. With a system-size independent cutoff, the number of factors that have to be considered for each move remains constant. These factors can be identified with a standard cell system and evaluated in $\mathcal{O}(1)$ operations. However, the essential long-range nature of the potential is then sacrificed.

Strategies have been devised to improve the $\mathcal{O}(N)$ scaling of the Metropolis algorithm natively. One natural approach exploits the decision-problem nature of the Metropolis filter, where the Bernoulli distribution giving rise to Eq. (4) is sampled with a uniform random number $\mathtt{ran}(0,1) = \Upsilon$, and the move is accepted if $\Upsilon < \exp\left(-\beta\Delta U\right)$ and rejected otherwise. For a given dynamical cutoff r_c, the accept/reject decision may be taken on the basis of $\Delta U^{\rm short}(r_c)$, the change of energy with contributions up to a pair distance r_c if we can establish rigorous r_c-dependent bounds,

$$e^{-\beta\Delta U^{\rm short}} - \epsilon^- < e^{-\beta\Delta U} < e^{-\beta\Delta U^{\rm short}} + \epsilon^+. \tag{5}$$

The neglected long-range contribution $\Delta U^{\rm long}(r_c)$ does not change the accept/reject decision if the random number Υ is outside the window of Eq. (5) (see Fig. 2). The cutoff r_c is then increased if required, adding particles in concentric layers around $\mathbf{x}$ or $\mathbf{x}'$. Under mild conditions on the maximum local density, this approach can be made rigorous.[27] However, typical Monte Carlo simulations require an enormous number of accept/reject decisions to be made (their number can easily reach 10^{10} or 10^{12}). This, in essence, makes it difficult to implement dynamical cutoffs in practice. In addition, dynamical cutoffs do not work for long-range potentials which, in d-dimensional space, decay as r^{-d} or slower, as the contribution of $U^{\rm long}$ then remains large for all values of r_c.[23,33] This is because the accept/reject decision is then typically taken at dynamical cutoffs r_c comparable to the system size.

In the multi-time-step Metropolis algorithm,[28] the Lennard-Jones system again splits the potential U into long-range and short-range contributions as indicated in Eq. (2), performing a series of n_s provisional moves on the basis of $U^{\rm short}$ only. This amounts to proposing a move of $k < N$ particles (with coordinates $\mathbf{Y}$, see Fig. 3) from $\mathbf{Y}^{\rm old}$ to $\mathbf{Y}^{\rm new}$. A sequence of n_s "short-range" moves are then accepted/rejected with the Metropolis filter for $U^{\rm long}$ only. Calling for simplicity the original configuration $\mathbf{X}$ (with the k particles at

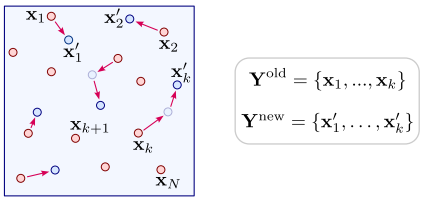

FIG. 3. Multi-step Metropolis algorithm. In Algorithm 1, moves are first made with U^{short} only. Initial and final positions are stored. The proposed move $\mathbf{Y}^{\text{old}} \to \mathbf{Y}^{\text{new}}$ is subject to the Metropolis filter with U^{long}.

$\mathbf{Y}^{\text{old}}$) and the target configuration $\mathbf{X}'$ (with the k particles at $\mathbf{Y}^{\text{new}}$), the composite probabilities are

$$\begin{aligned} p(\mathbf{X} \to \mathbf{X}') &= p^{\text{short}}(\mathbf{X} \to \mathbf{X}') \min\left[1, \frac{\pi^{\text{long}}(\mathbf{X}')}{\pi^{\text{long}}(\mathbf{X})}\right] \\ p(\mathbf{X}' \to \mathbf{X}) &= p^{\text{short}}(\mathbf{X}' \to \mathbf{X}) \min\left[1, \frac{\pi^{\text{long}}(\mathbf{X})}{\pi^{\text{long}}(\mathbf{X}')}\right]. \end{aligned} \tag{6}$$

Taking the ratio in Eq. (6) and noting that p^{short}, as a sequence of n_s reversible moves, satisfies the detailed-balance condition with respect to π^{short},

$$\pi^{\text{short}}(\mathbf{X}) p^{\text{short}}(\mathbf{X} \to \mathbf{X}') = \pi^{\text{short}}(\mathbf{X}') p^{\text{short}}(\mathbf{X}' \to \mathbf{X}),$$

we see that the full transition matrix satisfies detailed balance with respect to the full Boltzmann distribution $\pi = \pi^{\text{short}}\pi^{\text{long}}$,

$$\pi(\mathbf{X}) p(\mathbf{X} \to \mathbf{X}') = \pi(\mathbf{X}') p(\mathbf{X}' \to \mathbf{X}), \tag{7}$$

so that the algorithm is correct [see Algorithm 1 (`multi-step-metropolis`) for an implementation].

In Algorithm 1 (`multi-step-metropolis`), short-range moves can be implemented in $\mathcal{O}(1)$ with an appropriate cell system (Ref. 21, Sec. 2.4.1). The construction of the sets $\mathbf{Y}^{\text{old}}$ and $\mathbf{Y}^{\text{new}}$ is also $\mathcal{O}(1)$ per element so that the complexity of the entire short-range loop is $\mathcal{O}(n_s)$. However, the complexity of the long-range decision in Algorithm 1 is $\mathcal{O}(n_s \times N)$, with an acceptance probability that plummets with increasing n_s, a parameter that should therefore be chosen to be as small as possible. This leads to $n_s = 1$, and this defeats the initial intention.

Algorithm 1 (`multi-step-metropolis`), in spite of its serious limitations, illustrates the split of U into $U^{\text{short}}(r)$ and $U^{\text{long}}(r)$ that effectively leads to a factorization of the interaction. The concept will be extended in the following sections, where it leads to native $\mathcal{O}(1)$ long-range algorithms. The split of the interactions also echoes Hamiltonian Monte Carlo,[35–37] where provisional moves $\mathbf{X} \to \mathbf{X}'$ are proposed from a sequence of molecular-dynamics iterations rather than from a number of Metropolis steps. In Hamiltonian Monte Carlo, accumulated finite-time-step errors are also eliminated by a Metropolis filter, and the Boltzmann distribution is again sampled without approximations. This works because the errors are small, but it fails for large long-range systems because the errors are extensive in N even for $n_s = 1$.

ALGORITHM 1. `multi-step-metropolis`. Composite iteration of the algorithm of Ref. 28. The calculation of $U^{\text{short}}_{\mathbf{x}}$ is $\mathcal{O}(1)$ with the use of a grid. The prime in $\sum'$ eliminates double counting of pairs $(\mathbf{x}, \mathbf{x}')$ and $(\mathbf{x}', \mathbf{x})$. For large n_s, this algorithm has many long-range rejections. See the Appendix for a Python script for the two-dimensional Lennard-Jones system.

procedure `multi-step-metropolis`
input $\mathbf{X}$ (configuration at time t)
$\mathbf{Y}^{\text{old}} \leftarrow \emptyset$; $\mathbf{Y}^{\text{new}} \leftarrow \emptyset$
for $i = 1, \ldots, n_s$: (short-range steps)
$\mathbf{x} \leftarrow$ **choice**$(\mathbf{X})$ (random particle)
$U^{\text{short}}_{\mathbf{x}} \leftarrow \sum_{\mathbf{x}'' \in \mathbf{X}\setminus\{\mathbf{x}\}:|\mathbf{x}''-\mathbf{x}|<r_c} U(|\mathbf{x}'' - \mathbf{x}|)$
$\mathbf{x}' \leftarrow \mathbf{x} + \Delta\mathbf{x}$ (with $|\Delta\mathbf{x}| < \delta$)
$U^{\text{short}}_{\mathbf{x}'} \leftarrow \sum_{\mathbf{x}'' \in \mathbf{X}\setminus\{\mathbf{x}\}:|\mathbf{x}''-\mathbf{x}'|<r_c} U(|\mathbf{x}'' - \mathbf{x}'|)$
$\Upsilon \leftarrow$ `ran`$(0,1)$
if $\Upsilon < \exp\left[-\beta\left(U^{\text{short}}_{\mathbf{x}'} - U^{\text{short}}_{\mathbf{x}}\right)\right]$:
$\mathbf{X} \leftarrow \{\mathbf{x}'\} \cup \mathbf{X} \setminus \{\mathbf{x}\}$; $\mathbf{Y}^{\text{new}} \leftarrow \mathbf{Y}^{\text{new}} \cup \{\mathbf{x}'\}$
if $\mathbf{x} \notin \mathbf{Y}^{\text{new}}$: $\mathbf{Y}^{\text{old}} \leftarrow \mathbf{Y}^{\text{old}} \cup \{\mathbf{x}\}$
else : $\mathbf{Y}^{\text{new}} \leftarrow \mathbf{Y}^{\text{new}} \setminus \{\mathbf{x}\}$
$\Delta U^{\text{long}} \leftarrow \sum'_{\substack{\mathbf{x}\in\mathbf{Y}^{\text{new}},\mathbf{x}'\in\mathbf{X} \\ |\mathbf{x}-\mathbf{x}'|>r_c}} U(|\mathbf{x}-\mathbf{x}'|) - \sum'_{\substack{\mathbf{x}\in\mathbf{Y}^{\text{old}},\mathbf{x}'\in\mathbf{Y}^{\text{old}}\cup\mathbf{X}\setminus\mathbf{Y}^{\text{new}} \\ |\mathbf{x}-\mathbf{x}'|>r_c}} U(|\mathbf{x}-\mathbf{x}'|)$
$\Upsilon \leftarrow$ `ran`$(0,1)$ (long-range decision)
if $\Upsilon > \exp\left(-\beta\Delta U^{\text{long}}\right)$:
$\mathbf{X} \leftarrow \mathbf{Y}^{\text{old}} \cup \mathbf{X} \setminus \mathbf{Y}^{\text{new}}$
output $\mathbf{X}$ (configuration at time $t+1$)

B. Factorized Metropolis filter

The multi-time-step Metropolis approach of Algorithm 1 separates the interactions into two factors, one short-range and one long-range. More generally, total energies U can often be written as

$$U(\mathbf{X}) = \sum_{M \in \mathcal{M}} U_M, \tag{8}$$

with factors M in a set $\mathcal{M}$ so that the Boltzmann distribution is

$$\pi(\mathbf{X}) = \exp\left[-\beta U(\mathbf{X})\right] = \prod_{M \in \mathcal{M}} \exp\left(-\beta U_M\right). \tag{9}$$

We can then replace the Metropolis filter of Eq. (4), written as

$$\mathcal{P}^{\text{Met}}(\mathbf{x} \to \mathbf{x}') = \min\left[1, \prod_{M \in \mathcal{M}} \exp\left(-\beta\Delta U_M\right)\right], \tag{10}$$

with the factorized Metropolis filter[24]

$$\mathcal{P}^{\text{fact}}(\mathbf{x} \to \mathbf{x}') = \prod_{M \in \mathcal{M}} \underbrace{\min\left[1, \exp\left(-\beta\Delta U_M\right)\right]}_{\mathcal{P}^{\text{fact}}_M(\mathbf{x}\to\mathbf{x}')}. \tag{11}$$

That the factorized filter also satisfies the detailed-balance condition can be shown just as in Eqs. (6) and (7).

ALGORITHM 2. `factorized-metropolis`. Proposing a move and accepting it in case of consensus via Eq. (11), samples the Bolzmann distribution of Eq. (9). See the Appendix for a Python script.

procedure `factorized-metropolis`
input $\mathbf{X}$ (configuration at time t)
$\mathbf{x} \leftarrow$ **choice**$(\mathbf{X})$ (random particle)
$\mathbf{x}' \leftarrow \mathbf{x} + \Delta\mathbf{x}$ (with $|\Delta\mathbf{x}| < \delta$)
for $\mathbf{x}'' \in \mathbf{X} \setminus \{\mathbf{x}\}$:
  $\Upsilon \leftarrow$ `ran`$(0,1)$
  if $\Upsilon > \exp\left[-\beta(U_{\mathbf{x}''\mathbf{x}'} - U_{\mathbf{x}''\mathbf{x}})\right]$: **goto** 1
$\mathbf{X} \leftarrow \{\mathbf{x}'\} \cup \mathbf{X} \setminus \{\mathbf{x}\}$
1 **output** $\mathbf{X}$ (configuration at time $t+1$)

In our context, the use of pair factors $M = \{i,j\}$ with $U_M = U(|\mathbf{x}_i - \mathbf{x}_j|)$ is natural (even though the concept is easily generalizable[24,25,33]). A proposed move of a single particle then involves $N-1$ factors (that is, $N-1$ pairs). It is accepted by consensus, that is, if all factors accept it [see Algorithm 2 (`factorized-metropolis`) and Fig. 4; see Ref. 25 for a general discussion of the consensus principle]. As written, Algorithm 2 (`factorized-metropolis`) has complexity $\mathcal{O}(N)$ per move, corresponding to the number of evaluations of the pair potential U. We will show in the following how to implement the algorithm by only evaluating $\mathcal{O}(1)$ pair potentials per move.

C. Bounding the potential, cell-veto algorithm

The factorized Metropolis filter replaces the accept/reject decision based on the total energy $U(\mathbf{x})$ with independent decisions for all factor potentials U_M with $M \in \mathcal{M}$. This replacement carries enormous potential for speedup as, for a given distant pair of particles $\mathbf{x}$ and $\mathbf{x}''$, we need not systematically evaluate all the rejection probabilities,[23] applying a principle called "thinning."[26] Heuristically, let us suppose that, at a distance between $\mathbf{x}$ and $\mathbf{x}''$, the veto probability $q = 1 - \exp(-\beta\Delta U_M)$ can be bounded by $\epsilon \ll 1$. Rather than check the factor at each proposed move, we check it with probability ϵ (on average every $1/\epsilon$ times). Only when we check it, we evaluate q, and we veto the move with probability q/ϵ. Because of $q = \epsilon \times (q/\epsilon)$, the overall veto probability is correct, but the bound has allowed us to drastically reduce the number of evaluations.

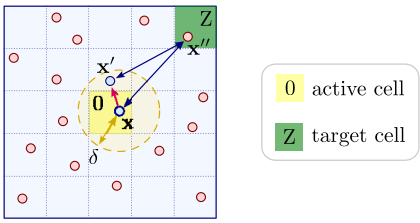

FIG. 5. For a pair of far-away particles $\mathbf{x}$ and $\mathbf{x}''$, the factor rejection probability for a move $\mathbf{x} \to \mathbf{x}'$ can be bounded by a constant q_Z depending on the target cell Z relative to the active cell 0.

The cell-veto algorithm[23] puts the above heuristic into practice by dividing the periodic simulation box into cells that usually contain at most one particle (see Fig. 5). Rather than the particle $\mathbf{x}''$, one addresses a target cell Z relative to the active cell 0 containing the active particle $\mathbf{x}$. The upper bound q_Z for the rejection plays the role of ϵ in the heuristic. It satisfies,

$$q_Z \geq \max_{\substack{\mathbf{x}\in 0, \mathbf{x}''\in Z \\ \mathbf{x}':|\mathbf{x}'-\mathbf{x}|<\delta}} \{1 - \exp\left[-\beta(U_{\mathbf{x}''\mathbf{x}'} - U_{\mathbf{x}''\mathbf{x}})\right]\}. \tag{12}$$

To yield the correct rejection probability $[1 - \exp[-\beta(U_{\mathbf{x}''\mathbf{x}'} - U_{\mathbf{x}''\mathbf{x}})]]$, a veto called by cell Z, with probability q_Z, must be confirmed with probability

$$\frac{[1 - \exp\left[-\beta(U_{\mathbf{x}''\mathbf{x}'} - U_{\mathbf{x}''\mathbf{x}})\right]]}{q_Z}. \tag{13}$$

Two complications must be solved. First, a cell Z neighboring the active cell 0 may not allow for a finite upper bound q_Z. Second, the cell Z may contain more than one particle. Both complications can be solved, with the second case requiring a set of "surplus" particles. The "two-pebble" decision,[25] first for the cell Z, then for the particle $\mathbf{x}'' \in$ Z, is illustrated in Algorithm 3 (`cell-veto`), which also implements a first loop over neighbor and surplus particles, as in Algorithm 2 (`factorized-metropolis`), and a cell-veto loop over

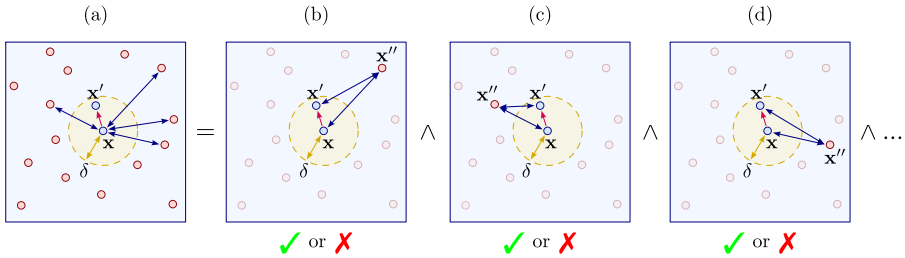

FIG. 4. Factorized Metropolis filter using the consensus principle (see Algorithm 2). (a) The active particle $\mathbf{x}$ may interact with all other particles. (b), (c), (d), . . . Every factor of the factorized Metropolis filter in Eq. (11) independently decides whether to veto a proposed move. The conjunctions $\wedge$ denote the consensus: the move $\mathbf{x} \to \mathbf{x}'$ is accepted if no factor vetoes it.

ALGORITHM 3. `cell-veto`. Naïve cell-veto algorithm. The far-away cells Z use a two-pebble veto, one for the cell, one for the particle [see patch in Algorithm 4, which can reduce the complexity per move from $\mathcal{O}(N)$ to $\mathcal{O}(1)$]. See the Appendix for a Python script.

procedure `cell-veto`
input $\mathbf{X}$ (configuration of N particles)
$\mathbf{x} \leftarrow$ **choice**$(\mathbf{X})$
$\mathbf{x}' \leftarrow \mathbf{x} + \Delta\mathbf{x}$ (with $|\Delta\mathbf{x}| < \delta$)
for $\mathbf{x}'' \in \{\text{neighbor}\} \cup \{\text{surplus}\}$: [see Fig. 6(b)]
  $\Upsilon \leftarrow$ `ran`$(0,1)$
  if $\Upsilon > \exp\left[-\beta(U_{\mathbf{x}''\mathbf{x}'} - U_{\mathbf{x}''\mathbf{x}})\right]$: **goto** 1
for Z $\in \mathcal{F}$, non-empty : (loop over far-away cells $\mathcal{F}$)
  $\Upsilon_1 \leftarrow$ `ran`$(0,1)$
  if $\Upsilon_1 < q_Z$:
    $\Upsilon_2 \leftarrow$ `ran`$(0,1)$
    $\mathbf{x}'' \leftarrow$ particle in cell Z
    if $\Upsilon_2 < \dfrac{1 - \exp\left[-\beta(U_{\mathbf{x}''\mathbf{x}'} - U_{\mathbf{x}''\mathbf{x}})\right]}{q_Z}$: **goto** 1
$\mathbf{X} \leftarrow \{\mathbf{x}'\} \cup \mathbf{X} \setminus \{\mathbf{x}\}$
1 **output** $\mathbf{X}$

far-away cells that form a set $\mathcal{F}$ (see Fig. 6). This loop over the elements of $\mathcal{F}$ is "naïve": it gives the correct result but is inefficient.

III. CELL-VETO LOOP

The loop over the far-away cells $\mathcal{F}$ in the naïve Algorithm 3 (`cell-veto`) scales linearly with N. The linear scaling can be avoided by sampling the subset $\mathcal{S}_{\text{veto}}$ of cells vetoing the proposed move (Sec. III A). Two different scenarios are possible. For infinitesimal moves, when the cell-veto algorithm is implemented for a continuous-time Markov process, as in the event-chain Monte Carlo algorithm,[38] the veto probabilities are themselves infinitesimal and, most of the time, the set $\mathcal{S}_{\text{veto}}$ is empty. Otherwise, if $\mathcal{S}_{\text{veto}} \neq \emptyset$, this set contains a single cell, which can be sampled in $\mathcal{O}(1)$ using Walker's algorithm[39] (Sec. III B). In contrast, for finite moves, the set $\mathcal{S}_{\text{veto}}$ may contain more than one element, in which case the set can also be sampled without iterating over all cells.[29] We present an algorithm that reduces the sampling problem to the infinitesimal-move case (Sec. III C).

A. The set $\mathcal{S}_{\text{veto}}$ of veto cells

In Algorithm 3 (`cell-veto`), all cells are looped over, but the veto must be confirmed in a second step only for the subset $\mathcal{S}_{\text{veto}} \subset \mathcal{F}$ of vetoing cells. The "cell-accept" decisions are confirmed automatically.[25] Clearly, the probability of any such subset is

$$\pi(\mathcal{S}_{\text{veto}}) = \underbrace{\prod_{Z \in \mathcal{S}_{\text{veto}}} q_Z}_{\text{veto}} \underbrace{\prod_{Z \notin \mathcal{S}_{\text{veto}}} (1 - q_Z)}_{\text{no veto}} . \tag{14}$$

The sample space for which $\pi(\mathcal{S}_{\text{veto}})$ is a probability measure is the powerset of $\mathcal{F}$, the set of its subsets. Given $\mathcal{S}_{\text{veto}}$, the loop over far-away cells becomes superfluous, and the cell-veto algorithm can be patched as in Algorithm 4.

B. Infinitesimal moves, event-chain Monte Carlo

In the non-reversible event-chain Monte Carlo algorithm,[24,38,40] the move

$$\mathbf{x} \to \mathbf{x}' = \mathbf{x} + \mathbf{v}\mathrm{d}t, \tag{15}$$

is infinitesimal, and it is repeated until vetoed by a factor M (in our case, a pair). The particle originally at $\mathbf{x}$ thus moves in continuous time along a ray indicated by the velocity $\mathbf{v}$ in Eq. (15). This velocity can often be taken of unit norm $|\mathbf{v}| = 1$, and the inverse move need not be proposed. After the veto, the other element in the pair factor becomes the active particle, and this situation can be easily generalized to larger factors describing, for example, many-body interactions or groups of atoms inside a pair of molecules.[30,31,41,42]

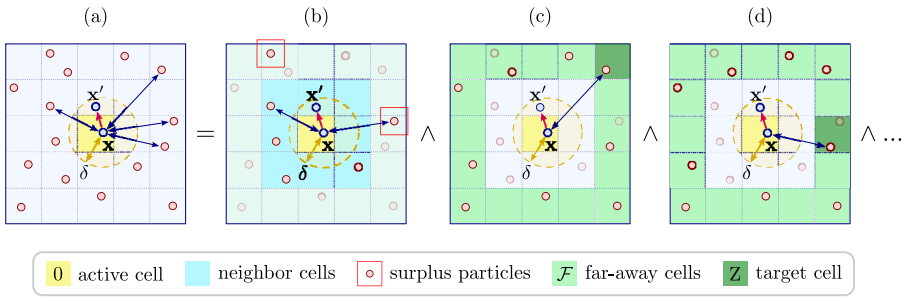

FIG. 6. Naïve cell-veto algorithm for a move $\mathbf{x} \to \mathbf{x}'$, as implemented in Algorithm 3 (`cell-veto`). (a) The box is divided into cells, which rarely contain more than one particle. (b) Neighbor and surplus particles are handled directly, as in Algorithm 2. (c), (d), . . . Cell-veto loop. The far-away cells $\mathcal{F}$ are iterated over. Each cell Z, then the particle $\mathbf{x}''$ inside it, may veto the move.

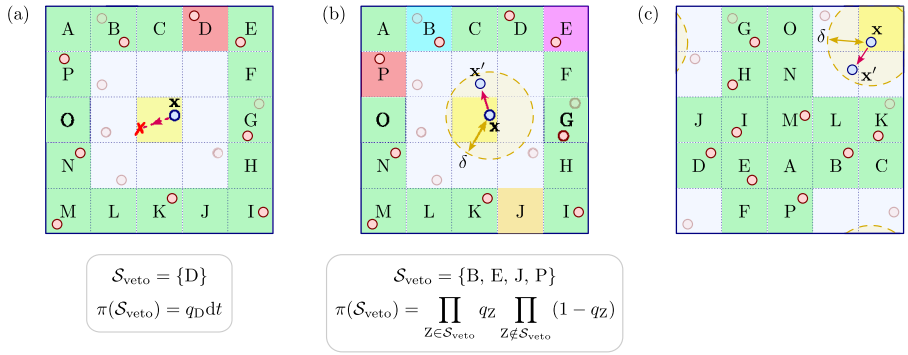

FIG. 7. Subsets of cells. (a) An infinitesimal move, vetoed by at most a single cell (see Sec. III B). (b) A finite move $\mathbf{x} \to \mathbf{x}'$, vetoed by a non-trivial subset of cells (see Sec. III C). (c) With periodic boundary conditions, the set $\mathcal{F}$ is defined relative to the active cell.

ALGORITHM 4. `cell-veto(patch)`. Patch of Algorithm 3. The iteration over the far-away cells $\mathcal{F}$ is replaced by the iteration over the vetoing cells $\mathcal{S}_{\rm veto} \subset \mathcal{F}$. This algorithm, with an efficient sampling of $\mathcal{S}_{\rm veto}$, is implemented in our "$\mathcal{O}(1)$ per move" Python scripts of the Appendix.

procedure `cell-veto(patch)`
...
(treating neighbor and surplus particles as in Alg. 3)
...
$\mathcal{S}_{\rm veto} \leftarrow$ sampled from $\pi(\mathcal{S}_{\rm veto})$ (see Eq. (14))
for Z $\in \mathcal{S}_{\rm veto}$, non-empty : (patch of cell-veto loop)
$\Upsilon \leftarrow$ `ran(0,1)`
$\mathbf{x}'' \leftarrow$ (unique) particle $\in$ Z
if $\Upsilon < \dfrac{1-\exp\left[-\beta(U_{\mathbf{x}''\mathbf{x}'} - U_{\mathbf{x}''\mathbf{x}})\right]}{q_{\rm Z}}$: **goto** 1
$\mathbf{X} \leftarrow \{\mathbf{x}'\} \cup \mathbf{X} \setminus \{\mathbf{x}\}$
1 **output X**

Within the active cell "0," a large number of subsequent infinitesimal moves along the ray are not vetoed by any cell, with the probability

$$\underbrace{1 - \sum_{\rm Z\in\mathcal{F}} q_{\rm Z}}_{q_{\mathcal{F}}}, \tag{16}$$

giving rise to a waiting-time distribution

$$\mathbb{P}(t)\mathrm{d}t = q_{\mathcal{F}} \exp\left(-q_{\mathcal{F}} t\right)\mathrm{d}t. \tag{17}$$

This waiting time for the next cell event can be sampled as $t = \log\left[\mathtt{ran}(0,1)\right]/q_{\mathcal{F}}$. At the event time, the cell responsible for the veto can be sampled in $\mathcal{O}(1)$ using Walker's algorithm, as is illustrated in Figs. 7 and 8 for far-away cells $\mathcal{F} = \{\rm A, \ldots, P\}$. In a periodic simulation box, the Walker table concerns relative cell locations, and it is prepared at the beginning of the simulation and then translates together with the active particle [see Fig. 7(c)]. The method outlined earlier can be integrated into Algorithm 4 for a native $\mathcal{O}(1)$ event-chain Monte Carlo algorithm for the Lennard-Jones model and many other models. See Fig. 10 for run-time data and the Appendix for a Python script. Some implementation details of the event-chain Monte Carlo algorithm underlying the Python script are not described here but are contained in Refs. 23, 33, 38, and 40.

C. Finite displacements

For a finite proposed move $\mathbf{x} \to \mathbf{x}'$, in discrete time and with finite cell-veto probabilities $q_{\rm Z}$ for Z $\in \mathcal{F}$, more than one cell can veto, leading to a non-trivial subset $\mathcal{S}_{\rm veto}$. While the cell-accepts are definite, the vetoes are not. They must be confirmed by a second "pebble." The set $\mathcal{S}_{\rm veto}$, with the probability distribution given in Eq. (14), can be sampled naïvely by iterating over the elements of $\mathcal{F}$.

To sample the set $\mathcal{S}_{\rm veto}$ more efficiently, we consider a Poisson process of intensity $\lambda_{\rm Z}$ in the time interval [0,1], with a distribution of the number of events as

$$\mathbb{P}(n \text{ events}) = \frac{\lambda_{\rm Z}^n}{n!} \mathrm{e}^{-\lambda_{\rm Z}}. \tag{18}$$

If we identify $q_{\rm Z}$ with the probability that one or more events take place in the time interval [0,1], we have

$$q_{\rm Z} = \mathbb{P}(\geq 1 \text{ events}), \tag{19}$$

$$= 1 - \mathbb{P}(0 \text{ events}) = 1 - \mathrm{e}^{-\lambda_{\rm Z}}, \tag{20}$$

which gives $\lambda_{\rm Z} = -\log(1-q_{\rm Z})$. The waiting-time distribution of this Poisson process is

$$\mathbb{P}(t)\mathrm{d}t = \lambda_{\rm Z} \exp\left(-\lambda_{\rm Z} t\right)\mathrm{d}t, \tag{21}$$

and events from this process can be sampled as $t = \log\left[\mathtt{ran}(0,1)\right]/\lambda_{\rm Z}$. Naïvely, we might loop over the cells in $\mathcal{F}$, translate the

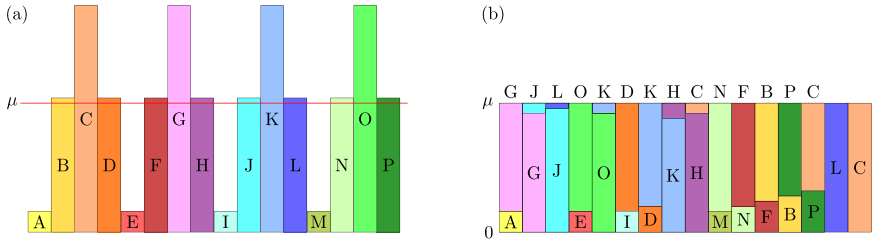

FIG. 8. Walker's algorithm for sampling the subset $\mathcal{S}_{\text{veto}} \subset \mathcal{F} = \{\text{A}, \text{B}, \ldots, \text{P}\}$ of vetoing cells (see Fig. 7). The probabilities shown correspond to $\{q_\text{A}, \ldots, q_\text{P}\}$ for infinitesimal moves or $\{\lambda_\text{A}, \ldots, \lambda_\text{P}\}$ for finite moves. (a) The probabilities reflect the spatial symmetry of the cells. (b) The probabilities are rearranged into a rectangle, with at most two elements stacked on top of each other. After a $\mathcal{O}(|\mathcal{F}|)$ preparation [from (a) to (b)], each sample takes two random numbers, so is $\mathcal{O}(1)$for$|\mathcal{F}| \to \infty$.

ALGORITHM 5. `poisson-veto`. Sampling the set $\mathcal{S}_{\text{veto}}$ through $|\mathcal{F}|$ Poisson processes. See the Appendix for a Python script and its patch in Algorithm 6 for a much faster version with a single Poisson process.

procedure `poisson-veto`
input $\{q_\text{A}, \ldots, q_\text{P}\}$ (q_Z : veto probability of cell Z)
$\mathcal{S}_{\text{veto}} \leftarrow \emptyset$
for Z $\in \{\text{A}, \ldots, \text{P}\}$:
  $\lambda_\text{Z} \leftarrow -\log(1 - q_\text{Z})$
  $t \leftarrow -\log[\texttt{ran}(0,1)]/\lambda_\text{Z}$
  if $t < 1$: $\mathcal{S}_{\text{veto}} \leftarrow \mathcal{S}_{\text{veto}} \cup \{\text{Z}\}$
output $\mathcal{S}_{\text{veto}}$

cell-veto probabilities q_Z into the intensities λ_Z of associated Poisson processes, and check whether the waiting times are smaller than 1 [see Algorithm 5 (`poisson-veto`)]. Integrated into Algorithm 4 (`cell-veto(patch)`), this exactly reproduces the factorized Metropolis algorithm.

As the Poisson processes in Algorithm 5 (`poisson-veto`) are independent, one may run them in parallel (see Fig. 9). A single

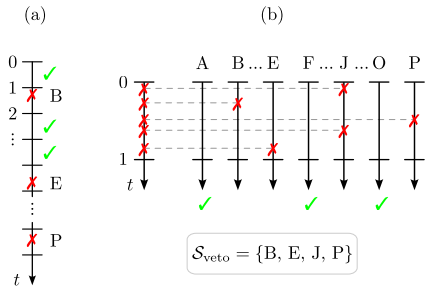

FIG. 9. Sampling the veto cells $\mathcal{S}_{\text{veto}}$ [see Fig. 7(b)]. (a) Using subsequent Poisson processes for $\{\text{A}, \ldots, \text{P}\}$ [see Algorithm 5 (`poisson-veto`)]. (b) Using a single Poisson process, together with Walker's algorithm. Each cell may veto repeatedly [see Algorithm 6 (`poisson-veto(patch)`)].

ALGORITHM 6. `poisson-veto(patch)`. Efficient sampling of the set $\mathcal{S}_{\text{veto}}$ of veto cells through a single Poisson processes. See the Appendix for a Python script producing output equivalent to Algorithm 5 (`poisson-veto`) and Algorithm 6 (`poisson-veto(patch)`) and for a Python script complementing Algorithm 4 (`cell-veto(patch)`) for an efficient simulation of the two-dimensional Lennard-Jones system.

procedure `poisson-veto(patch)`
input $\{q_\text{A}, \ldots, q_\text{P}\}$ (q_Z : veto probability of cell Z)
$\{\lambda_\text{A}, \ldots, \lambda_\text{P}\} \leftarrow \{-\log(1 - q_\text{A}), \ldots, -\log(1 - q_\text{P})\}$
$\lambda \leftarrow \sum_{\text{Z} \in \mathcal{F}} \lambda_\text{Z}$ (total intensity of all Poisson processes)
$\mathcal{S}_{\text{veto}} \leftarrow \emptyset$
$t \leftarrow 0$
while True :
  $t \leftarrow t - \log[\texttt{ran}(0,1)]/\lambda$ (waiting time to next event)
  if $t > 1$: **break**
  Z $\leftarrow$ `walker`$(\{\lambda_\text{A}, \ldots, \lambda_\text{P}\})$
  $\mathcal{S}_{\text{veto}} \leftarrow \mathcal{S}_{\text{veto}} \cup \{\text{Z}\}$
output $\mathcal{S}_{\text{veto}}$

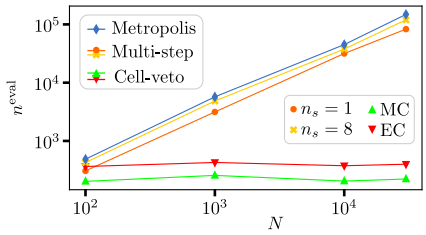

FIG. 10. Number n^{eval} of pair-potential evaluations per unit distance traveled by one particle in the two-dimensional Lennard-Jones system at constant number density $\rho = 0.05$, with respect to the system size. The results were obtained using the Python scripts in the Appendix. The lines are guides to the eye, yet the constant scaling of the cell-veto algorithms is evident ("MC" denotes the reversible variant of the factorized Metropolis algorithm, and "EC" the event-chain Monte Carlo algorithm).

Poisson process with intensity $\lambda = \sum_{Z \in \mathcal{F}} \lambda_Z$ then controls the time at which one of the $|\mathcal{F}|$ Poisson processes, that is, one of the far-away cells, holds a veto. Walker's algorithm can again be used to identify this cell [see Algorithm 6 (`poisson-veto(patch)`)]. This algorithm avoids the iteration over the set $\mathcal{F}$ of far-away cells, and as Walker's algorithm is of complexity $\mathcal{O}(1)$, the complexity of our algorithm is essentially $\mathcal{O}(|\mathcal{S}_{\text{veto}}|)$. The time interval of the Poisson process may be adjusted so that $\mathcal{S}_{\text{veto}}$ contains only few elements. Algorithm 6 (`poisson-veto(patch)`) can be integrated into Algorithm 4 for a native Monte Carlo algorithm for the Lennard-Jones model (see Fig. 10 for run-time data and the Appendix for a Python implementation).

IV. CONCLUSIONS

In this paper, we have discussed the role of long-range potentials within molecular simulation and, especially, within the Monte Carlo approach to sampling. In a situation where enormous effort has been dedicated to the efficient evaluation of the total energy U and its gradient, we point out that Monte Carlo sampling of the Boltzmann distribution $\exp(-\beta U)$ does not generally require knowledge of U, in other words, its evaluation. The cell-veto algorithms that we presented here use a "two-pebble" decision that is akin to the thinning of Poisson processes, and they can be extended in many directions. In related research,[30,33,34] we have shown that the SPC/Fw water model frequently used in biomolecular simulations can be sampled exactly in the canonical ensemble without relying on thermostats, discretization, cutoffs, and grids, with the same complexity $\mathcal{O}(1)$ per event that we have discussed in this paper in the simple setting of pair interactions and the two-dimensional Lennard-Jones system. It remains to be seen whether our native methods can be applied more generally to molecular applications and whether they can solve the problems posed by truncating, smoothing out, or discretizing interactions in long-time simulations. In our examples, the long-range nature of the potentials relates to the physics of the photon (rather than the meson) for the Coulomb interaction, and the London dispersion force for the Lennard-Jones potential is of great importance for the physics of interfaces and surfaces in soft condensed matter.

ACKNOWLEDGMENTS

This research was supported by a grant from the Simons Foundation (Grant No. 839534, MET). We acknowledge N. Bou-Rabee, K. Hukushima, T. Schlick, Y. Sugita, and S. Vionnet for their helpful discussions.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Gabriele Tartero: Data curation (equal); Investigation (equal); Software (equal); Validation (equal); Visualization (lead); Writing – original draft (equal); Writing – review & editing (equal). **Werner Krauth**: Conceptualization (lead); Project administration (lead); Supervision (lead); Validation (equal); Writing – original draft (lead); Writing – review & editing (lead).

DATA AVAILABILITY

The algorithms and computer programs discussed in this paper are publicly available at https://github.com/jellyfysh/MCLongRange.git.

APPENDIX: COMPUTER PROGRAMS

This paper is accompanied by the `MCLongRange` software package, which is published as an open-source project under the GNU GPLv3 license. `MCLongRange` is available on GitHub as a part of the JeLLyFysh organization. The package contains Python scripts for many algorithms discussed in this paper and used to produce the numerical results of Fig. 10. Scripts implement the Metropolis algorithm, the multi-time-step Metropolis algorithm, the factorized Metropolis algorithm, and the two versions of the cell-veto algorithm described in Sec. III B (non-reversible event-chain Monte Carlo) and in Sec. III C (reversible Monte Carlo). See Table I for the correspondence between Python scripts, pseudocode algo-

TABLE I. Correspondence between the Python scripts of the `MCLongRange` software package and the pseudocode algorithms presented in this paper.

Python script	Pseudocode algorithm	Theoretical discussion
metropolis.py	···	Section II A
multi-step_metropolis.py	Algorithm 1 (`multi-step-metropolis`)	Section II A and Ref. 28
factorized_metropolis.py	Algorithm 2 (`factorized-metropolis`)	Section II B and Ref. 24
MC_cell-veto.py	Algorithm 4 (`cell-veto(patch)`), with the set $\mathcal{S}_{\text{veto}}$ sampled using Algorithm 6 (`poisson-veto(patch)`)	Sections II C and III C
EC_cell-veto.py	Non-reversible version of Algorithm 4 (`cell-veto(patch)`), with the set $\mathcal{S}_{\text{veto}}$ sampled using Walker's algorithm	Sections II C and III B, Refs. 23, 33, 38, and 40
poisson_veto.py	Algorithm 5 (`poisson-veto`) Algorithm 6 (`poisson-veto(patch)`)	Section III C

rithms, and our theoretical discussions. The url of the repository is https://github.com/jellyfysh/MCLongRange.git.

REFERENCES

[1] T. Schlick, *Molecular Modeling and Simulation: An Interdisciplinary Guide* (Springer, 2002).
[2] H. Eissfeller and M. Opper, "New method for studying the dynamics of disordered spin systems without finite-size effects," Phys. Rev. Lett. **68**(13), 2094–2097 (1992).
[3] J. Jordan, R. Orús, G. Vidal, F. Verstraete, and J. I. Cirac, "Classical simulation of infinite-size quantum lattice systems in two spatial dimensions," Phys. Rev. Lett. **101**(25), 250602 (2008).
[4] K. Van Houcke, E. Kozik, N. Prokof'ev, and B. Svistunov, Phys. Procedia **6**, 95 (2010).
[5] K. Van Houcke, F. Werner, E. Kozik, N. Prokof'ev, B. Svistunov, M. J. Ku, A. T. Sommer, L. W. Cheuk, A. Schirotzek, and M. W. Zwierlein, "Feynman diagrams versus Fermi-gas Feynman emulator," Nat. Phys. **8**, 366 (2012).
[6] R. Rossi, "Determinant diagrammatic Monte Carlo algorithm in the thermodynamic limit," Phys. Rev. Lett. **119**, 045701 (2017).
[7] M. Levitt, "Birth and future of multiscale modeling for macromolecular systems (Nobel Lecture)," Angew. Chem., Int. Ed. **53**, 10006–10018 (2014).
[8] J. Cardy, *Scaling and Renormalization in Statistical Physics* (Cambridge University Press, 1996).
[9] D. Aldous and P. Diaconis, "Shuffling cards and stopping times," Am. Math. Mon. **93**(5), 333–348 (1986).
[10] J. G. Propp and D. B. Wilson, "Exact sampling with coupled Markov chains and applications to statistical mechanics," Random Struct. Algorithms **9**(1–2), 223–252 (1996).
[11] A. J. Stone, *The Theory of Intermolecular Forces*, 2nd ed. (Oxford University Press, 2013).
[12] D. E. Shaw, M. M. Deneroff, R. O. Dror, J. S. Kuskin, R. H. Larson, J. K. Salmon, C. Young, B. Batson, K. J. Bowers, J. C. Chao, M. P. Eastwood, J. Gagliardo, J. P. Grossman, C. R. Ho, D. J. Ierardi, I. Kolossváry, J. L. Klepeis, T. Layman, C. McLeavey, M. A. Moraes, R. Mueller, E. C. Priest, Y. Shan, J. Spengler, M. Theobald, B. Towles, and S. C. Wang, "Anton, a special-purpose machine for molecular dynamics simulation," in *Proceedings of the 34th Annual International Symposium on Computer Architecture, ISCA '07* (ACM, New York, NY, USA, 2007), pp. 1–12.
[13] D. E. Shaw, P. Maragakis, K. Lindorff-Larsen, S. Piana, R. O. Dror, M. P. Eastwood, J. A. Bank, J. M. Jumper, J. K. Salmon, Y. Shan, and W. Wriggers, "Atomic-level characterization of the structural dynamics of proteins," Science **330**(6002), 341–346 (2010).
[14] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations," J. Comput. Phys. **73**(2), 325–348 (1987).
[15] B. Smit and D. Frenkel, "Vapor–liquid equilibria of the two-dimensional Lennard-Jones fluid(s)," J. Chem. Phys. **94**(8), 5663–5668 (1991).
[16] B. Smit, "Phase diagrams of Lennard-Jones fluids," J. Chem. Phys. **96**(11), 8639–8640 (1992).
[17] Y. A. Lei, T. Bykov, S. Yoo, and X. C. Zeng, "The Tolman length: Is it positive or negative?," J. Am. Chem. Soc. **127**(44), 15346–15347 (2005).
[18] A. Tröster, M. Oettel, B. Block, P. Virnau, and K. Binder, "Numerical approaches to determine the interface tension of curved interfaces from free energy calculations," J. Chem. Phys. **136**(6), 064709 (2012).
[19] C. Tempra, O. H. S. Ollila, and M. Javanainen, "Accurate simulations of lipid monolayers require a water model with correct surface tension," J. Chem. Theory Comput. **18**(3), 1862–1869 (2022).
[20] Y. Yu, A. Krämer, R. M. Venable, B. R. Brooks, J. B. Klauda, and R. W. Pastor, "CHARMM36 lipid force field with explicit treatment of long-range dispersion: Parametrization and validation for phosphatidylethanolamine, phosphatidylglycerol, and ether lipids," J. Chem. Theory Comput. **17**(3), 1581–1595 (2021).
[21] W. Krauth, *Statistical Mechanics: Algorithms and Computations* (Oxford University Press, 2006).
[22] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov Chains and Mixing Times* (American Mathematical Society, 2008).
[23] S. C. Kapfer and W. Krauth, "Cell-veto Monte Carlo algorithm for long-range systems," Phys. Rev. E **94**, 031302 (2016).
[24] M. Michel, S. C. Kapfer, and W. Krauth, "Generalized event-chain Monte Carlo: Constructing rejection-free global-balance algorithms from infinitesimal steps," J. Chem. Phys. **140**(5), 054116 (2014).
[25] G. Tartero and W. Krauth, "Concepts in Monte Carlo sampling," Am. J. Phys. **92**(1), 65–77 (2024).
[26] P. A. W. Lewis and G. S. Shedler, "Simulation of nonhomogeneous Poisson processes by thinning," Nav. Res. Logist. Q. **26**(3), 403–413 (1979).
[27] F. Müller, H. Christiansen, S. Schnabel, and W. Janke, "Fast, hierarchical, and adaptive algorithm for Metropolis Monte Carlo simulations of long-range interacting systems," Phys. Rev. X **13**(3), 031006 (2023).
[28] B. Hetényi, K. Bernacki, and B. J. Berne, "Multiple 'time step' Monte Carlo," J. Chem. Phys. **117**(18), 8203–8207 (2002).
[29] M. Michel, X. Tan, and Y. Deng, "Clock Monte Carlo methods," Phys. Rev. E **99**, 010105 (2019).
[30] P. Höllmer, A. C. Maggs, and W. Krauth, "Fast, approximation-free molecular simulation of the SPC/Fw water model using non-reversible Markov chains," Sci. Rep. **14**(1), 16449 (2024).
[31] J. Harland, M. Michel, T. A. Kampmann, and J. Kierfeld, "Event-chain Monte Carlo algorithms for three- and many-particle interactions," Europhys. Lett. **117**(3), 30001 (2017).
[32] J. E. Lennard-Jones, "Cohesion," Proc. Phys. Soc. **43**(5), 461 (1931).
[33] M. F. Faulkner, L. Qin, A. C. Maggs, and W. Krauth, "All-atom computations with irreversible Markov chains," J. Chem. Phys. **149**(6), 064113 (2018).
[34] P. Höllmer, L. Qin, M. F. Faulkner, A. C. Maggs, and W. Krauth, "JELLYFYSH-Version1.0—A Python application for all-atom event-chain Monte Carlo," Comput. Phys. Commun. **253**, 107168 (2020).
[35] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid Monte Carlo," Phys. Lett. B **195**, 216–222 (1987).
[36] R. M. Neal, "MCMC using Hamiltonian dynamics," in *Handbook of Markov Chain Monte Carlo*, edited by S. Brooks, A. Gelman, G. Jones, and X.-L. Meng (Chapman and Hall/CRC, 2011), pp. 113–162.
[37] N. Bou-Rabee and J. M. Sanz-Serna, "Geometric integrators and the Hamiltonian Monte Carlo method," Acta Numer. **27**, 113–206 (2018).
[38] W. Krauth, "Event-chain Monte Carlo: Foundations, applications, and prospects," Front. Phys. **9**, 229 (2021).
[39] A. J. Walker, "An efficient method for generating discrete random variables with general distributions," ACM Trans. Math. Softw. **3**(3), 253–256 (1977).
[40] E. P. Bernard, W. Krauth, and D. B. Wilson, "Event-chain Monte Carlo algorithms for hard-sphere systems," Phys. Rev. E **80**, 056704 (2009).
[41] M. Klement and M. Engel, "Efficient equilibration of hard spheres with Newtonian event chains," J. Chem. Phys. **150**(17), 174108 (2019).
[42] M. Michel, A. Durmus, and S. Sénécal, "Forward event-chain Monte Carlo: Fast sampling by randomness control in irreversible Markov chains," J. Comput. Graph. Stat. **29**(4), 689–702 (2020).