Computer Physics Communications 219 (2017) 246–254

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

A derivation and scalable implementation of the synchronous parallel kinetic Monte Carlo method for simulating long-time dynamics

Hye Suk Byun [a], Mohamed Y. El-Naggar [a,b,c], Rajiv K. Kalia [a,d,e,f], Aiichiro Nakano [a,b,d,e,f,*], Priya Vashishta [a,d,e,f]

[a] Department of Physics & Astronomy, University of Southern California, Los Angeles, CA 90089-0242, USA
[b] Department of Biological Sciences, University of Southern California, Los Angeles, CA 90089-0242, USA
[c] Department of Chemistry, University of Southern California, Los Angeles, CA 90089-0242, USA
[d] Department of Computer Science, University of Southern California, Los Angeles, CA 90089-0242, USA
[e] Department of Chemical Engineering & Materials Science, University of Southern California, Los Angeles, CA 90089-0242, USA
[f] Collaboratory for Advanced Computing and Simulations, University of Southern California, Los Angeles, CA 90089-0242, USA

ARTICLE INFO

Article history:
Received 3 February 2017
Received in revised form 23 April 2017
Accepted 31 May 2017
Available online 27 June 2017

Keywords:
Kinetic Monte Carlo simulation
Divide-and-conquer algorithm
Parallel computing

ABSTRACT

Kinetic Monte Carlo (KMC) simulations are used to study long-time dynamics of a wide variety of systems. Unfortunately, the conventional KMC algorithm is not scalable to larger systems, since its time scale is inversely proportional to the simulated system size. A promising approach to resolving this issue is the synchronous parallel KMC (SPKMC) algorithm, which makes the time scale size-independent. This paper introduces a formal derivation of the SPKMC algorithm based on local transition-state and time-dependent Hartree approximations, as well as its scalable parallel implementation based on a dual linked-list cell method. The resulting algorithm has achieved a weak-scaling parallel efficiency of 0.935 on 1024 Intel Xeon processors for simulating biological electron transfer dynamics in a 4.2 billion-heme system, as well as decent strong-scaling parallel efficiency. The parallel code has been used to simulate a lattice of cytochrome complexes on a bacterial-membrane nanowire, and it is broadly applicable to other problems such as computational synthesis of new materials.

1. Introduction

Ever-increasing processing power of parallel computers [1] is continuously extending the spatiotemporal scales of particle simulations [2], where each particle could represent an atom in molecular dynamics (MD) simulations or a human in agent-based social simulations. Essential to extending the spatial scale is linear-scaling algorithms based on spatial locality principles, in which the computational complexity scales linearly with the number of particles N [3–6]. A harder problem is to increase the time scale of processes that can be simulated, due to the inherently sequential nature of time as a result of causality [7]. In some cases, temporal locality principles alleviate the sequential-time bottleneck in MD simulations [2,4,6,8]. Namely, a many-particle system tends to stay near a local minimum-energy configuration over a long duration of time, which is bounded by a rare transition over short time period to another minimum. In such a case, the transition state theory (TST) [9,10] uses a local equilibrium assumption to reformulate the sequential long-time dynamics as computationally more efficient parallel search for low-activation transition events [11–14], where the rates of the events are computed from their activation barriers.

The most widely used simulation method based on TST is kinetic Monte Carlo (KMC) [15–20]. In KMC simulations, an event to occur is stochastically selected from a database of events. The simulated time progresses according to Poisson statistics, where the time increment t at each KMC step is inversely proportional to the sum of the rates of all possible events. Since the summed rate grows as at least $O(N)$, KMC simulations progress much more slowly for larger systems, *i.e.*, $t = O(1/N)$. With an $O(N)$ implementation of the computation for a single KMC step, therefore, the computational complexity to simulate a physical time duration of τ is $O(N \times \tau/t) = O(N \times \tau N) = O(N^2)$. The conventional KMC algorithm is thus not scalable for large N. Here, it should be noted that single-step KMC computation can be performed in $O(\log N)$ time instead of $O(N)$ [21], or even faster if the types of possible events are bounded as $O(1)$ [22].

More efficient long-time simulation is possible because of the spatiotemporal locality of activated events. Namely, they are usually localized not only temporally but also spatially [23]. This leads

* Corresponding author at: Collaboratory for Advanced Computing and Simulations, University of Southern California, Los Angeles, CA 90089-0242, USA. Fax: +1 213 821 2664.
E-mail address: anakano@usc.edu (A. Nakano).

http://dx.doi.org/10.1016/j.cpc.2017.05.028
0010-4655/© 2017 Elsevier B.V. All rights reserved.

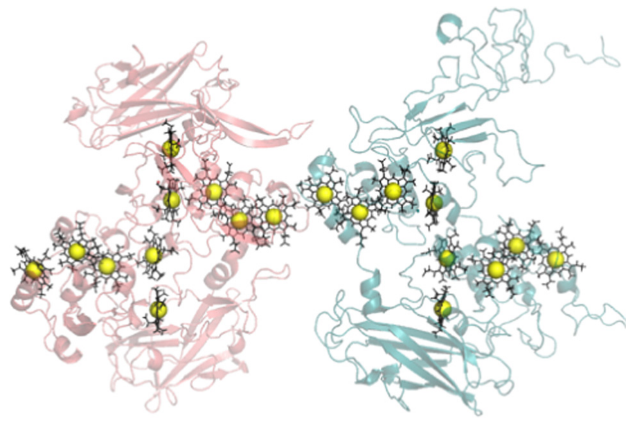

Fig. 1. Heme groups in a dimer of decaheme cytochromes, where the Fe atom within each heme is represented by a yellow sphere. Each of the two cytochromes (colored magenta and cyan) contains 10 hemes.

to computationally more efficient simulation methods that concurrently sample multiple, spatially localized events [24–28]. In such cases, event sampling may be enhanced by an ensemble-mean field approach. For example, a time-dependent Hartree (TDH) approximation has been employed to sample the dynamics of a small key subsystem within a large molecule [29]. Here, an ensemble of MD simulations for the subsystem is embedded in an MD simulation of the entire molecule, where the subsystem and the rest of the molecule interact via an ensemble-mean field [29]. In the light of the localized nature of atomistic events [23], we here introduce a local transition-state (LTS) approximation, in conjunction with the TDH approximation, into the KMC simulation method. This leads to a divide-and-conquer strategy that simulates multiple events concurrently based on spatial decomposition. The resulting simulation method is equivalent to the synchronous parallel KMC (SPKMC) method [25–28]. Furthermore, we introduce a dual linked-list cell (DL^{2}C) method to further reduce the computational cost. This paper is organized as follows. Section 2 describes a derivation and implementation of the SPKMC algorithm. Numerical results are presented in Section 3, and Section 4 contains conclusions.

2. Methods

In this section, we present a new derivation of the synchronous parallel kinetic Monte Carlo (SPKMC) algorithm and its implementation on parallel computers using a dual linked-list cell (DL^{2}C) method. As a concrete example, we use electron-transfer (ET) dynamics between heme groups in a large network of cytochromes [30,31]. (Heme is an organic compound called porphyrin that contains an iron (Fe) atom at its center.) Fig. 1 shows a dimer of decaheme cytochromes and the heme groups in it. The Fe atom in each heme can exist in either of the two valence states, Fe^{2+} or Fe^{3+}. Conversion of irons between Fe^{2+} and Fe^{3+} allows for the hopping of electrons between adjacent hemes. KMC simulation treats electron-hopping events in a network of N hemes, where a heme site at position $\mathbf{q}_i$ is labeled by index $i \in \{1, \ldots, N\}$. The ith heme is either occupied by an electron ($n_i = 1$ or reduced, corresponding to Fe^{2+}) or unoccupied ($n_i = 0$ or oxidized, corresponding to Fe^{3+}), where n_i is the electron occupation number of the ith heme. The system dynamics are characterized by (i) electron hopping rates W_{ji} from the ith heme to the jth heme for adjacent (i, j) pairs, (ii) electron injection rate W_{red} into a selected entrance heme i_{ent}, and (iii) electron-ejection rate W_{ox} from an exit heme i_{exit}. In our work, W_{ji} is computed from the positions of hemes, $\mathbf{q}_i$ and $\mathbf{q}_j$, and their precomputed free energies, G_i and G_j [30]. In addition, W_{ji} depends on the occupation numbers, n_i and n_j, since an electron can hop from i to j only when $n_i = 1$ and $n_j = 0$. Due to the exponential decay of the electron-hopping rate with respect to the heme-pair distance, $W_{ji} = 0$ when $|\mathbf{q}_i - \mathbf{q}_j| > q_{\mathrm{cut}}$, where $q_{\mathrm{cut}} \sim 1$ nm is a cutoff distance. The following discussion also applies to other applications such as photoexcitation dynamics in solar cells [20] and computational synthesis of new materials based on chemical vapor decomposition and other techniques [19], as long as an event is spatially localized within a cutoff distance.

2.1. Kinetic Monte Carlo simulation

To introduce a notation necessary for the derivation of the SPKMC algorithm, Appendix A describes the conventional KMC simulation method [15–20]. We initialize a KMC simulation by emptying all heme sites and resetting the time to 0. At each KMC step, one of the following events occurs: (i) an electron is injected with rate W_{red} if the entrance heme is unoccupied; (ii) an electron is ejected with rate W_{ox} if the exit heme is occupied; or (iii) an electron hops from heme i to one of its nearest-neighbor hemes, j, with rate W_{ji} if heme i is occupied and heme j is unoccupied. The method for calculating the rates for the ET dynamics can be found in Ref. [30]. KMC simulation consists of a time-stepping loop. Let e be one of the possible events listed above, with W_e being its rate, E be the total number of possible events, and

$$W = \sum_{e=1}^{E} W_e \tag{1}$$

be the sum of the rates of all possible events. At each KMC step, the time is incremented by

$$t = -\ln(\xi_1)/W, \tag{2}$$

where ξ_1 is a uniform random number in the range [0,1]. The probability of choosing a particular event is proportional to its rate, and specific event e^* is chosen such that

$$e^* = \min_e \left\{ \sum_{c=1}^{e} W_c > W\xi_2 \right\}, \tag{3}$$

where ξ_2 is another uniform random number in [0,1].

2.2. Synchronous parallel KMC algorithm

The standard KMC method in Section 2.1 is not scalable to larger system sizes. The cumulative event rate W grows as $O(N)$, and accordingly the time scale of the simulation determined by its inverse becomes progressively smaller in larger systems, as is seen in Eq. (2). To overcome this scaling problem, we parallelize KMC simulations in a divide-and-conquer (DC) fashion, using a synchronous formulation and graph coloring to avoid conflicting events [25–28]. To do so, we first introduce a local transition-state (LTS) approximation, in which events outside a cutoff distance are assumed to be statistically independent. We then introduce a time-dependent Hartree (TDH) approximation, *i.e.*, the simulated system is subdivided into spatially localized domains and local events in a domain are sampled independently of those in the other domains. Appendix B provides a formal derivation of the resulting SPKMC algorithm.

Domain decomposition: The SPKMC algorithm partitions the 3-dimensional space $\Re^3$ into spatially localized domains $\Re^3_d$ that are mutually exclusive,

$$\Re^3 = \bigcup_d \Re^3_d; \quad \Re^3_d \cap \Re^3_{d'} = \varnothing. \tag{4}$$

For simplicity, we consider a simple mesh decomposition, in which the total rectangular space is subdivided into domains of equal

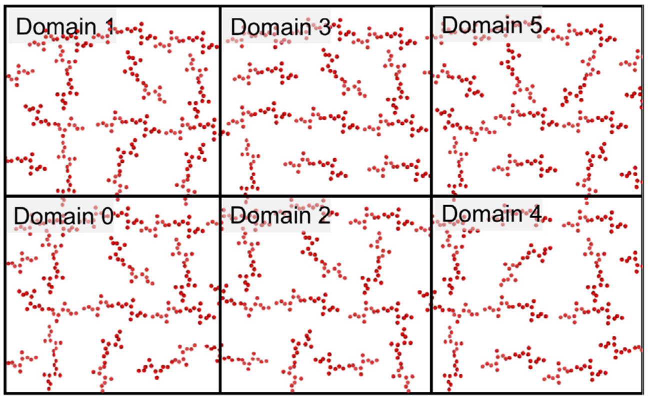

Fig. 2. Two-dimensional schematic of domain decomposition into an array of 3×2 domains in the x and y directions. Each cytochrome in a lattice of cytochrome dimers is represented by its 10 Fe positions (10 red spheres).

volume. The domains are arranged in a cubic lattice. Fig. 2 shows a two-dimensional schematic of the domain decomposition.

In SPKMC, the cumulative rate $W^{(d)}$ is computed for each non-overlapping domain $\Re_d^3$ as in Eq. (1) but summing only over the local events within the dth domain $\Re_d^3$. Below, we denote the rate of the eth event in domain d as $W_e^{(d)}$. Then, the global maximum rate is computed as $W_{\max} = \max_d \left\{W^{(d)}\right\}$. SPKMC simulation is performed concurrently among all the domains similarly to the sequential KMC algorithm in Section 2.1, except that a null event (where nothing occurs) is generated with the rate of $W_0^{(d)} = W_{\max} - W^{(d)}$. This allows globally synchronous time evolution, where the time is incremented by $t = -\ln(\xi_1)/W_{\max}$ at each KMC step, where ξ_1 is a uniform random number in [0,1] which is common to all the domains. By keeping the size of domains constant and increasing the number of domains, $W_{\max}$ should remain nearly constant and thus $t = O(1)$ instead of $O(1/N)$ in the conventional KMC method. (A more rigorous mathematical analysis is required for the actual scaling of $W_{\max}$ with respect to the system size [32].) Each domain d then generates a local random number $\xi_2^{(d)}$ $(\in [0, 1])$ independently of each other, and chooses an event to occur, $e^{(d)}$, according to

$$e^{(d)} = \min_e \left\{ \sum_{c=1}^{e} W_c^{(d)} > W_{\max}\xi_2^{(d)} \right\}. \tag{5}$$

To avoid conflicting events within q_{cut} to occur between neighboring domains, the domains are colored so that no domains of the same color are adjacent to each other. In a cubic lattice in three dimensions, this is achieved by arranging adjacent domains into a block of $2 \times 2 \times 2$ domains. Here, we assume an even number of domains in each Cartesian direction. We require that the side length of each domain to be larger than $2q_{\text{cut}}$, so that no heme can be a destination of more than one electron hopping event. Within each block, the eight domains are indexed with an octant index (or color), $c \in [0, \ldots, 7]$. Fig. 3(a) shows a two-dimensional schematic of the blocked domain decomposition. In each SPKMC simulation step, a color is chosen randomly, and events occur only in the domains of the chosen color.

2.3. Dual linked-list cell (DL^{2}C) algorithm

A naive double-loop implementation to compute electron hopping rates W_{ji} between heme pairs (i, j) would scale as $O(N^2)$. In fact, with a finite cutoff length q_{cut}, each heme interacts with only a limited number of other hemes, $(4\pi N/3V)q_{\text{cut}}^3 \sim O(1)$, on average ($V$ is the volume of the system). This makes the computational cost

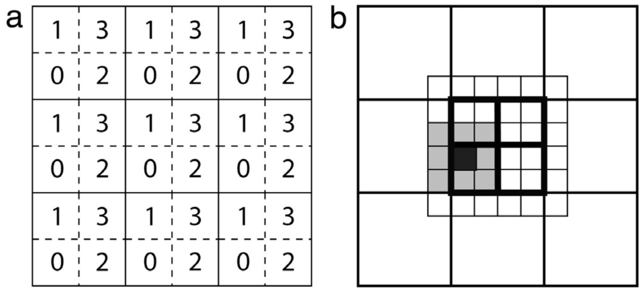

Fig. 3. (a) Two-dimensional schematic of the blocked domain decomposition in SPKMC. Each block of domains is delineated by solid lines, and is subdivided into 2×2 quadrants labeled (or colored) 0, 1, 2 or 3. (b) Caching of heme occupancy in parallel SPKMC with 3×3 spatial subsystems. The central subsystem is a block of 2×2 domains as delineated by thick lines. Each small square is a cell slightly larger than q_{cut}, used to compute electron-hopping rates between neighboring hemes. A heme in a dark gray cell only interacts with those in the nearest neighbor cells colored in light gray. Accordingly, the central subsystem only needs to be augmented with one layer of cells with the heme information in them cached from the neighbor subsystems.

to process all pairs of neighbor hemes to be $O(N)$. The linked-list cell algorithm computes the entire pair interaction with $O(N)$ operations [33,34]. This algorithm first divides the system into small cells of equal volume, where the edge length of each cell is made at least q_{cut}. A heme in a cell interacts with only other hemes in the same cell and its 26 neighbor cells. The hemes belonging to a cell are organized as a linked list. In an array implementation, *head*[c] holds the index of the first heme in the cth cell, or *head*[c] = NULL if there is no heme in the cell; *lscl*[i] holds the heme index to which the ith heme points. All hemes in cell c are traversed by following links in *lscl*[ ], starting with the heme specified by *head*[c]. The $O(N)$ algorithm loops over cells, and for each cell, it loops over the 27 neighbor cells (including itself). For each pair of neighbor cells, the corresponding linked lists are used to traverse all pairs of hemes residing in the cell pair.

For further improvement of performance, we have designed a DL^{2}C algorithm, which utilizes two types of linked-list cells: (1) small cells for constructing neighbor-heme lists for managing nearest-neighbor hopping events; (2) larger cells for domain-block coloring. In our original implementation, cell linked lists, *head*[ ] and *lscl*[ ], were used in two-fold purposes. First, they were used in function *make_list*( ) to construct neighbor-heme lists *lsngb*[ ][ ], before the main KMC simulation loop is entered. For the ith heme, *lsngb*[i][0] stores the number of other hemes within q_{cut} and *lsngb*[i][j] stores the ID of the jth neighbor heme. Construction of *lsngb*[ ][ ] amounts to doubly nested **for** loops over cells (each with a **while** loop to follow the link of hemes associated with the cell), which is computationally intensive. Second, the same cell linked lists were used at each KMC step for traversing the hemes in a given colored domain.

While the computation in the first usage scales as $O(N)$, the prefactor grows quadratically with the average number of hemes in a cell, or equivalently sixth power of the cell size, r_{cell}. To reduce the computation in the first usage, DL^{2}C uses the minimal cell size $r_{\text{cell}} \sim q_{\text{cut}} \sim 1$ nm, for constructing *lsngb*[ ][ ] in function *make_list*( ). Subsequently, the algorithm dynamically creates another set of cell linked lists using a larger r_{cell}, which is dictated by parallel-computing considerations discussed in the next subsection. Typical values used in sample applications in the next section are $r_{\text{cell}} = 8$ nm or larger.

2.4. Parallelization

Spatial decomposition: To parallelize the SPKMC algorithm, the physical system to be simulated is partitioned into subsystems of equal volume [34,35]. On a parallel computer, parallel processes are logically arranged according to the topology of these physical subsystems. Hemes that are located in a particular subsystem are assigned to the corresponding process. For simplicity, we use simple 3D mesh (or torus because of the periodic boundary conditions) decomposition. Subsystems are (and accordingly processes are logically) arranged in a 3D array of dimensions $P_x \times P_y \times P_z$. Each subsystem is a parallelepiped of size $L_x \times L_y \times L_z$. Each process is given a unique process ID, $p \in [0, P-1]$, where $P = P_xP_yP_z$ is the total number of processes. We also define a vector process ID, $\vec{p} = (p_x, p_y, p_z)$, where $p_x \in [0, P_x-1]$, $p_y \in [0, P_y-1]$ and $p_z \in [0, P_z-1]$. The sequential and vector ID's are related through the relation, $p = p_xP_yP_z + p_yP_z + p_z$. Using periodic boundary conditions, every subsystem has 26 neighbor subsystems that share either a corner, an edge or a face with it.

To map the SPKMC algorithm onto the spatial decomposition, we here adopt a simple scheme, in which each spatial subsystem is a $2 \times 2 \times 2$ block of domains for graph coloring as explained in Section 2.2. This leads to coarse-grained parallelism, resulting in high parallel efficiency according to the analysis in the next section. Also, it should be noted that the TDH approximation underlying SPKMC becomes exact in the large granularity limit (Appendix B). In SPKMC, each process computes the electron injection rates into the hemes within its subsystem (*i.e.*, resident hemes) as well as the electron ejection and hopping rates from its resident hemes. Each process then randomly selects one event within its subsystem and updates the heme occupation numbers $\{n_i\}$ accordingly.

In order to compute electron hopping rates W_{ji} between heme pairs within the cutoff distance, heme positions $\{\mathbf{q}_i\}$ and free energies $\{G_i\}$ in the 26 neighbor subsystems (or corresponding processes), which are located within q_{cut} from the subsystem periphery, are copied from the corresponding processes to "this" process in function *heme_copy*() (see line 1 in Table 1). Through message forwarding, message passing is completed only with the six face-sharing neighbor subsystems, thus minimizing the total message latency [35]. These cached hemes are appended after the resident hemes in the corresponding arrays. Note that $\{\mathbf{q}_i\}$ and $\{G_i\}$ are static quantities, and thus *heme_copy*() is called only once at the beginning of the program before the main KMC loop begins. Here, we adopt a single program multiple data (SPMD) convention, so that the pseudocode of the SPKMC program in Table 1 is executed concurrently in all processes.

Next, each process allocates cell linked lists, *head*[] and *lscl*[], and builds them using the positions of both resident and cached hemes, using the linked-list cell size that is slightly larger than q_{cut}. The linked lists are then used to construct adjacent-heme lists, *lsngb*[][], for the resident hemes. The adjacency lists, *lsngb*[][], will be used in the main KMC loop for computing electron hopping rates. Once *lsngb*[][] are constructed, *head*[] and *lscl*[] are deallocated and are rebuilt such that each cell (or domain) is now an octant of the spatial subsystem (or domain block). These lists will be used in the main KMC loop for performing colored synchronous parallel KMC updates. The above computations are carried out in function *make_list*() before the main KMC loop is entered (line 2 in Table 1). In our implementation, *make_list*() also computes a part of the electron hopping rates that is independent of heme occupancy.

After initializing heme occupation numbers and resetting the time to zero, the algorithm enters the main KMC loop (line 4 in Table 1) to iterate KMC simulation steps. At each KMC step, function *occ_copy*() is called (line 5 in Table 1) first to cache boundary hemes' occupancies within q_{cut} from the periphery of the nearest neighbor processes (or spatial subsystems), so that the electron hopping rates for the resident hemes can be calculated locally independent of the other processes, reflecting the latest heme occupancies (line 6 in Table 1). Fig. 3(b) shows a schematic of this heme-occupancy caching operation.

Subsequently, the master process, $p = 0$, randomly selects an octant (or color) c and broadcasts it to all processes. Each process p then computes the sum of all rates within the selected octant, $W^{(p)}$, and its global maximum W_{max} over all processes is computed using a global reduction operation (lines 8 and 9 in Table 1). Using W_{max}, each process randomly selects an event and updates heme occupancies accordingly (line 10 in Table 1). Subsequently, the master process increments the time according to Poisson statistics (line 11 in Table 1).

After the heme occupancies are updated, electron hopping from some resident hemes may have changed the occupancy of a cached heme out of the subspace boundary. Each process sends the change of occupancies of cached hemes to their resident processes in function *occ_move*() (line 12 in Table 1).

Table 1 shows the SPKMC algorithm. The parallel SPKMC program is written in the C language, where interprocess communications are implemented using the message passing interface (MPI) library [36].

3. Results

The scalability of the algorithm has been tested on a cluster of multicore computers. This section first presents the scalability test results, followed by an example of its application to ET dynamics in a lattice of cytochromes on a bacterial-membrane nanowire.

3.1. Scalability tests

We test the scalability of the code on a parallel computer using ET dynamics in a lattice of decaheme cytochromes as an example. The scalability tests have been performed on a Linux cluster at the Center for High Performance Computing of the University of Southern California. Each computing node comprises dual octocore Intel Xeon central processing units (CPUs) with 64 GB DDR3 memory, operating at a clock cycle of 2.66 GHz. Each node thus has 16 CPU cores. These nodes are interconnected via 56.6 Gbit/s Infiniband network.

Weak scaling: We first perform isogranular-scaling tests for the code, in which the number of hemes per core N/P is kept constant. We compare three granularities, $N/P = 127{,}776$, 1,168,032 and 4,116,000, while varying the number of cores P from 16 to 1,024 for each granularity. Here, we use all 16 cores per computing node, and the number of nodes is varied from 1 to 256. The cutoff length of ET is $q_{\text{cut}} = 1$ nm, and the spatial subsystem size per core is a cube of side length $L = 88$ nm, 184 nm, and 280 nm for $N/P = 127{,}776$, 1,168,032, and 4,116,000, respectively. Fig. 4(a) shows the wall-clock time per KMC simulation step with scaled workloads. By increasing the number of hemes linearly with the number of cores, the wall-clock time remains almost constant, indicating excellent scalability. While slight increase of the wall-clock time for larger P is observed for the smallest granularity (*i.e.*, $N/P = 127{,}776$), such overhead is hardly visible for the largest granularity (*i.e.*, $N/P = 4{,}116{,}000$).

To quantify the parallel efficiency, we first define the speed of the code as a product of the total number of hemes and the number of KMC steps executed per second. The isogranular speedup is given by the ratio between the speed of P cores and that of 16 cores (*i.e.*, 1 computing node) as a reference system. The weak-scaling parallel efficiency is the isogranular speedup divided by $P/16$. Fig. 4(b) shows the measured weak-scaling parallel efficiency as a function of P for $N/P = 4{,}116{,}000$. With the granularity of 4,116,000 hemes per core, the parallel efficiency is 0.935 on $P = 1{,}024$ for a 4,214,784,000-heme system. This demonstrates a very high scalability of the code for multibillion-heme systems.

Table 1
Parallel SPKMC algorithm.

```
1.  heme_copy (): cache boundary-heme information within the cutoff distance $q_{cut}$ from the periphery of the nearest neighbor subsystems (or processes)
2.  make_list ()
    a. Allocate and build cell linked lists, head [] and lscl [], including both resident and cached hemes, with the linked-list cell size slightly larger than $q_{cut}$
    b. Make adjacent-heme lists lsngb [][] for the resident hemes using the cell linked lists
    c. Deallocate head [] and lscl []
    d. Allocate and build cell linked lists, head [] and lscl [], including both resident and cached hemes for the block of 8 domains
3.  Initialize the electron occupation numbers; reset the time $t \leftarrow 0$
4.  for step = 1 to Max_step
5.      occ_copy (): cache boundary hemes' occupancies from the neighbor processes
6.      Compute electron-hopping rates for the resident hemes
7.      Master process, $p = 0$, randomly selects an octant (or color) $c$ and broadcast it
8.      Compute the sum $W^{(p)}$ of the rates of events in octant $c$ ($p$ is the ID of "this" process)
9.      Global reduction: $W_{\max} = \max_p \{W^{(p)}\}$
10.     Pick an event $e^{(p)}$ according to Eq. (5) and update the heme occupation numbers accordingly
11.     Master process increments the time by $t = -\ln(\xi_1)/W_{\max}$
12.     occ_move (): send the change of occupancies of cached hemes to their resident processes
```

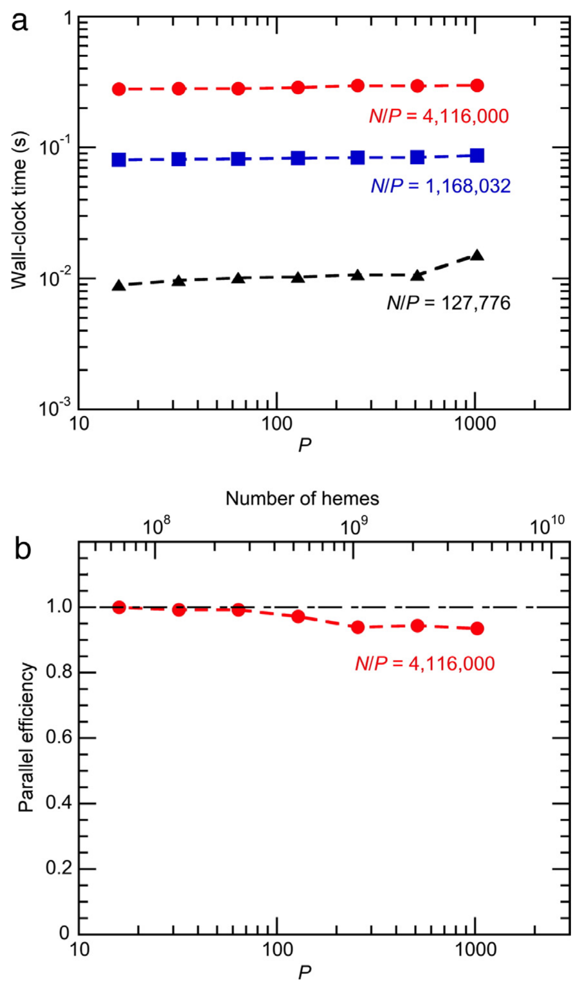

Fig. 4. (a) Wall-clock time per KMC simulation step, with scaled workloads, $N/P =$ 127,776 (black), 1,168,032 (blue) and 4,116,000 (red), as a function of the number of cores P ($P = 16, \ldots, 1{,}024$). (b) Weak-scaling parallel efficiency as a function of P for $N/P =$ 4,116,000 (red), compared with the ideal efficiency (black dash–dotted line).

The better parallel efficiency for the larger granularity observed above can be understood by a scalability analysis. Using the spatial decomposition and the $O(N)$ linked-list cell method, the parallel KMC simulation of N hemes executes independently on P processors, and the computation time is $T_{\text{comp}}(N, P) = aN/P$, where a is a constant. Here, we have assumed that the hemes are distributed uniformly on average, so that the average number of hemes per processor is N/P. The dominant overhead of the parallel KMC is heme caching, in which hemes near the subsystem boundary within a cutoff distance q_{cut} are copied from the nearest neighbor processors (line 5 in Table 1). Since this nearest-neighbor communication scales as the surface area of each spatial subsystem, its time complexity is $T_{\text{comm}}(N, P) = b(N/P)^{2/3}$, where b is a constant. Another major communication cost is for the global maximum computation to determine $W_{\max}$ (line 9 in Table 1) using the *MPI_Allreduce*(), which incurs $T_{\text{global}}(P) = c \log P$, where c is another constant.

The total execution time of the parallel MD program can thus be modeled as

$$\begin{aligned} T(N, P) &= T_{\text{comp}}(N, P) + T_{\text{comm}}(N, P) + T_{\text{global}}(P) \\ &= aN/P + b(N/P)^{2/3} + c \log P. \end{aligned} \tag{6}$$

For isogranular scaling, the number of atoms per processor, $N/P = n$, is constant, and the isogranular parallel efficiency is

$$\begin{aligned} E_P &= \frac{T(n, 1)}{T(nP, P)} = \frac{an}{an + bn^{2/3} + c \log P} \\ &= \frac{1}{1 + \frac{b}{a} n^{-1/3} + \frac{c}{an} \log P}. \end{aligned} \tag{7}$$

For a given number of processors, the efficiency E_P is larger for larger granularity n. For a given granularity, E_P is a weakly decreasing function of P, due to the very weak $\log P$ dependence.

Strong scaling: We also perform a strong-scaling test by simulating a cytochrome lattice containing a total of 3,145,728 hemes. In this test, the number of cores ranges from $P = 1$ to 16 on 1 computing node, while keeping the total problem size constant. In Fig. 5(a), the red circles show the wall-clock time per KMC simulation step as a function of P. The time-to-solution is reduced by a factor of 6.0 on 16 cores compared with the 1-core run. As in the weak-scaling test, the speed of the code is defined as a product of the total number of hemes and the number of KMC steps executed per second. The fixed problem-size (or strong-scaling) speedup is given by the ratio between the speed of P cores and that of 1 core. In Fig. 5(a), the blue squares show the speedup as a function of P, whereas the blue dash–dotted line is the ideal speedup. The strong-scaling speedup is 6.0, on 16 cores.

For fixed problem-size scaling, the global number of hemes, N, is fixed, and the speedup is given by

$$\begin{aligned} S_P &= \frac{T(N, 1)}{T(N, P)} = \frac{aN}{aN/P + b(N/P)^{2/3} + c \log P} \\ &= \frac{P}{1 + \frac{b}{a}\left(\frac{P}{N}\right)^{1/3} + \frac{c}{a}\frac{P \log P}{N}}, \end{aligned} \tag{8}$$

and the parallel efficiency is

$$E_P = \frac{S_P}{P} = \frac{1}{1 + \frac{b}{a}\left(\frac{P}{N}\right)^{1/3} + \frac{c}{a}\frac{P \log P}{N}}. \tag{9}$$

From this model, we can see that the efficiency is a decreasing function of P through both the $P^{1/3}$ and $P \log P$ dependences. This P dependence is much stronger than Eq. (7) for weak scaling. Consequently, it is more difficult to achieve high strong-scaling parallel efficiency compared with weak-scaling parallel efficiency. This is due to decreasing granularity, and accordingly increasing communication/computation ratio, for larger number of processors, in the former.

Though the strong-scaling speedup defined above is less than the ideal value in Fig. 5(a), using a larger number of computing cores (hence a larger number of spatial domains) increases the time simulated by KMC simulation. Namely, the average time increment per KMC step is inversely proportional to the sum of rates of all events within a domain, $W_{\max}$, which in turn is proportional to the number of hemes per domain. To quantify the actual speed of KMC simulation, we define a more practical measure of speed as a product of the total number of hemes and the simulated time by KMC per running time of the program. The simulated-time speedup is given by the ratio between the simulated-time speed of P cores and that of 1 core. Namely, the simulated-time speedup measures how fast the actual KMC simulation progresses on a parallel computer. In Fig. 5(b), the blue squares show the simulated-time speedup as a function of P. On 16 cores, KMC simulation progresses 95.9 times faster than on 1 core. This demonstrates a significant computational efficiency afforded by the divide-and-conquer approach.

3.2. *Application example*

Dissimilatory metal-reducing bacteria have evolved mechanisms to cope with living in anaerobic environments, allowing them to use abundant minerals outside the cell as respiratory electron acceptors, instead of oxygen or other soluble oxidants that would normally diffuse inside cells. This process is known as extracellular electron transfer (EET). *Shewanella oneidensis* MR-1 accomplishes EET by deploying multiheme cytochrome complexes that form 20–30 nm conduits through the periplasm and across the outer-membrane [37]. More recently, we learned that such cytochrome networks extend along micrometer-long membrane extensions called bacterial nanowires [38,39], and may even allow conduction over entire bacterial biofilms [40,41]. The study of such large-scale EET encompassing long length scales is beyond the scope of conventional KMC algorithms [30,31], and hence necessitates scalable parallel KMC simulations. The new parallel KMC simulation program has enabled us to perform KMC simulations with unprecedented spatiotemporal scales, for studying over a 1.12 μm long bacterial nanowire containing more than a thousand MtrC decaheme cytochromes assumed to be hexagonally and uniformly packed on the nanowire's membrane surface as a test case. Each MtrC molecule contains 10 hemes, thus a 12,800 heme network is formed in 1.12 μm long nanowire.

The nanowire's heme network is shown in Fig. 6(a). Electrons are injected into selected entrance hemes at one end of heme network with a rate of 10^7 s^{-1}, and are ejected with a rate of 10^7 s^{-1} from selected exit hemes at the other end of the network. The net electron flux through the whole nanowire is calculated as the slope of the number of electrons injected/ejected as a function of the total elapsed time after the system reaches a steady state (Fig. 6(b)). As a result of the simulation, we estimate $10^4 - 10^5$ s^{-1} of electron flux for the heme nanowire network. Remarkably, this rate matches previously measured ET rates to solid phase Fe(III) oxides from the MtrCAB complex assembled into proteoliposomes [42], and is enough to support a single cell's respiration rate [43]. With this basic demonstration of SPKMC to simulate a micrometer-long bacterial nanowire, future studies will allow us to perform additional mechanistic simulations that will reveal the effect of cytochrome density and orientation on nanowire ET rates, and even study larger systems such as the redox network of whole biofilms.

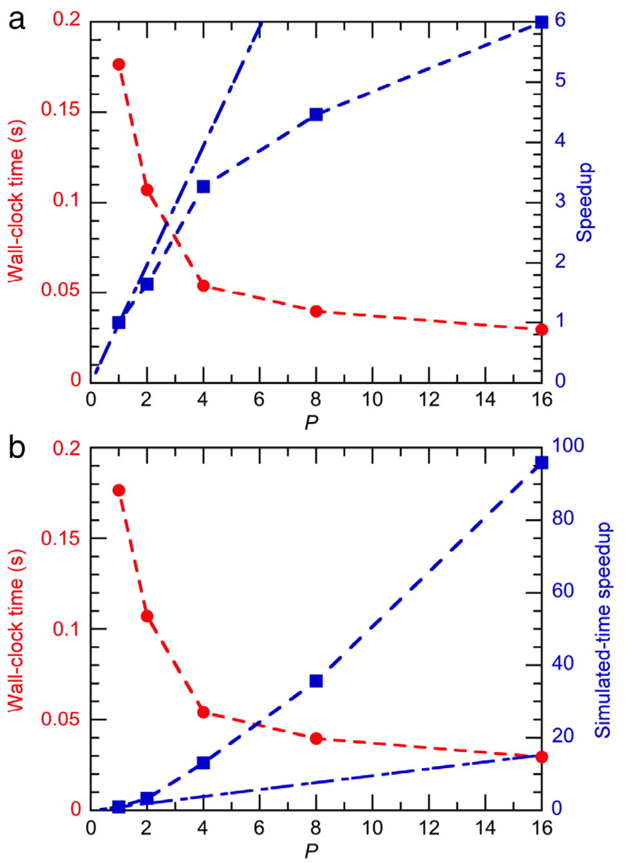

Fig. 5. (a) Wall-clock time per KMC simulation step (red) and speedup (blue) with strong scaling—3,145,728-heme system on P cores of Intel Xeon. The blue dash–dotted line shows the ideal speedup. (b) Wall-clock time per KMC simulation step (red) and simulated-time speedup (blue) with strong scaling—3,145,728-heme system on P cores of Intel Xeon. The blue dash–dotted line shows linear speedup.

4. Conclusions

We have designed a divide-and-conquer algorithm to simulate long-time dynamics of many-particle systems based on spatiotemporal locality principles. We provided a formal derivation of the known SPKMC algorithm based on local transition-state and time-dependent Hartree approximations. Such a formal derivation could serve as a starting point for developing new simulation methodologies. An example is the use of highly accurate descriptions of many-electron correlations within domains, which are coupled via simple electrostatics [44] in the algorithmic framework of divide-conquer-recombine (DCR) [6]. In addition, higher-order approximations could be introduced beyond the time-dependent Hartree approximation [45]. We then introduced a dual linked-list cell method to further reduce the computational cost. The resulting parallel KMC algorithm has achieved an excellent weak-scaling parallel efficiency and good strong-scaling parallel efficiency. The parallel KMC program has been used to simulate a lattice of cytochromes on a bacterial-membrane nanowire. Such long-time dynamics simulations are expected to shed light on many areas such as glass dynamics and biological self-assembly. Furthermore, the KMC algorithm is expected to play a major role in the computational synthesis of new materials based on chemical vapor deposition and other techniques.

Acknowledgments

The algorithmic development for computational synthesis was supported by the U.S. Department of Energy, Office of Science, Basic Energy Sciences, under Award # DE-SC0014607, as part of the

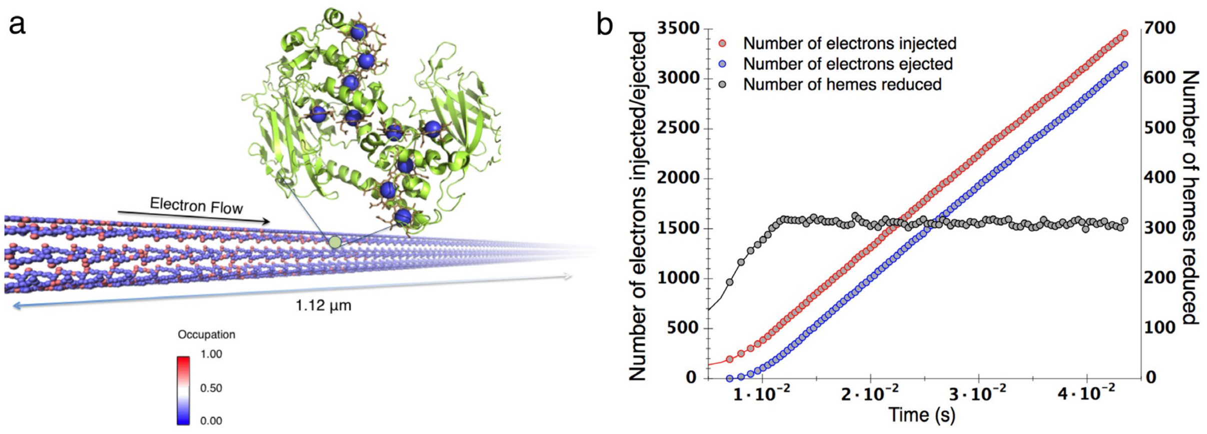

Fig. 6. (a) A simulation snapshot of the electron transport process along a 1.12 μm long bacterial nanowire containing 1,280 MtrC decaheme cytochromes (12,800 hemes). Electrons are injected from the nanowire's leftmost heme sites with injection rate of 10^7 s^{-1} and ejected at the rightmost heme sites with ejection rate of 10^7 s^{-1}. Occupation number of each heme site is color-coded. (b) Time evolution of the electron flux along the nanowire. The electron flux is defined as the number of electrons injected/ejected per unit time through an area perpendicular to the flux direction.

Computational Materials Science Program. Testing of the algorithm in microbial systems was supported by the Defense Threat Reduction Agency Grant # HDTRA1-14-1-0074. HSB and MYE-N were supported by the Division of Chemical Sciences, Geosciences, and Biosciences, Office of Basic Energy Sciences of the U.S. Department of Energy through grant DE-FG02-13ER16415. Numerical tests and simulations were performed at the Center for High Performance Computing of the University of Southern California.

Appendix A. Kinetic Monte Carlo (KMC) simulation method

For simplicity, we here consider a set of N particles following classical mechanics at positions $\mathbf{Q} \in \Re^{3N}$ with momenta $\mathbf{P} \in \Re^{3N}$, where $\Re$ is a set of real numbers. The dynamics of the system is specified by the Hamiltonian,

$$H(\mathbf{P}, \mathbf{Q}) = \frac{1}{2}\mathbf{P} \bullet \mathbf{M}^{-1} \bullet \mathbf{P} + V(\mathbf{Q}), \tag{A.1}$$

where $\mathbf{M} = \mathrm{diag}(m_1, m_1, m_1, \ldots, m_N, m_N, m_N) \in \Re^{3N\times 3N}$ is a diagonal mass matrix with m_i being the mass of the ith particle, and $V(\mathbf{Q})$ is the potential energy.

We consider a statistical ensemble of the system and introduce a probability density function $f(\mathbf{Q}, \mathbf{P}, t)$, such that $f(\mathbf{Q}, \mathbf{P}, t)d^{3N}\mathbf{Q}d^{3N}\mathbf{P}$ denotes the probability to find the system in a small $6N$-dimensional phase-space volume $d^{3N}\mathbf{Q}d^{3N}\mathbf{P}$ around coordinates $\mathbf{Q}$ and momenta $\mathbf{P}$ at time t [19,46]. Time evolution of $f(\mathbf{Q}, \mathbf{P}, t)$ is governed by the Liouville equation,

$$\frac{\partial f}{\partial t} = -\hat{L}f, \tag{A.2}$$

where $\hat{L}$ is the Liouville operator,

$$\hat{L} = \frac{\partial H}{\partial \mathbf{P}} \bullet \frac{\partial}{\partial \mathbf{P}} - \frac{\partial H}{\partial \mathbf{Q}} \bullet \frac{\partial}{\partial \mathbf{Q}} = \mathbf{M}^{-1} \bullet \mathbf{P} \bullet \frac{\partial}{\partial \mathbf{P}} + \mathbf{F}(\mathbf{Q}) \bullet \frac{\partial}{\partial \mathbf{Q}}, \tag{A.3}$$

and $\mathbf{F}(\mathbf{Q}) = -\partial V/\partial \mathbf{Q} \in \Re^{3N}$ are the interparticle forces.

Long-time dynamics of the system is often described in terms of a sequence of transitions between discrete states. Here, the total configuration space is partitioned into non-overlapping discrete states,

$$\Re^{3N} = \bigcup_{\alpha} \Re_{\alpha}^{3N}; \quad \Re_{\alpha}^{3N} \cap \Re_{\beta}^{3N} = \varnothing, \tag{A.4}$$

where $\forall \mathbf{Q} \in \Re_{\alpha}^{3N}$ converges (via a steepest-descent procedure) to the αth local minimum-energy configuration $\mathbf{Q}_{\alpha}^{\min}$, at which $\partial V/\partial \mathbf{Q}_{\alpha}^{\min} = 0$, and all the eigenvalues of the Hessian matrix, $\partial^2 V/\partial \mathbf{Q}^2|_{\mathbf{Q}=\mathbf{Q}_{\alpha}^{\min}}$, are positive. Let

$$P_{\alpha}(t) = \frac{1}{h^{3N}} \int_{\Re_{\alpha}^{3N}} d^{3N}\mathbf{Q} \int d^{3N}\mathbf{P} f(\mathbf{Q}, \mathbf{P}, t) \tag{A.5}$$

be the probability to find the system within $\Re_{\alpha}^{3N}$ at time t (h is the Planck constant). To determine the time evolution of $P_{\alpha}(t)$, we take the time derivative of Eq. (A.5) using the Liouville equation, Eq. (A.2). By applying Gauss' theorem, we obtain

$$\begin{aligned}\frac{dP_{\alpha}(t)}{dt} &= -\sum_{\beta} \frac{1}{h^{3N}} \int d^{3N}\mathbf{P} \int_{\partial\Re_{\beta\alpha}^{3N}} d\mathbf{s} \\ &\quad \bullet M^{-1} \bullet \mathbf{P}\Theta\left(d\mathbf{s} \bullet M^{-1} \bullet \mathbf{P}\right) f(\mathbf{Q}, \mathbf{P}, t) \\ &\quad + \sum_{\beta} \frac{1}{h^{3N}} \int d^{3N}\mathbf{P} \int_{\partial\Re_{\alpha\beta}^{3N}} d\mathbf{s} \\ &\quad \bullet M^{-1} \bullet \mathbf{P}\Theta\left(d\mathbf{s} \bullet M^{-1} \bullet \mathbf{P}\right) f(\mathbf{Q}, \mathbf{P}, t),\end{aligned} \tag{A.6}$$

where $d\mathbf{s} \in \Re^{3N}$ is an area-element vector normal to the surface, and the step function is defined as $\Theta(x) = 1\ (x \geq 0);\ 0\ (\text{else})$. In Eq. (A.6), we have partitioned the $(3N-1)$-dimensional surface, $\partial\Re_{\alpha}^{3N}$, of $\Re_{\alpha}^{3N}$ into

$$\partial\Re_{\alpha}^{3N} = \bigcup_{\beta} \partial\Re_{\beta\alpha}^{3N}, \tag{A.7}$$

where $\partial\Re_{\beta\alpha}^{3N}$ is the interface of $\Re_{\alpha}^{3N}$ and its neighbor state $\Re_{\beta}^{3N}$ with its normal vector pointing from $\Re_{\alpha}^{3N}$ to $\Re_{\beta}^{3N}$.

When the long-time dynamics consists of rare events (*i.e.*, if there is a separation of time scales between long time intervals between consecutive interstate transitions and rapid intrastate particle motions), the transition state theory (TST) [9,10] introduces a *local equilibrium approximation (LEA)*, *i.e.*, the phase-space density is in local equilibrium within each state:

$$f(\mathbf{Q}, \mathbf{P}, t) \approx \frac{P_{\alpha}(t)}{P_{\alpha}^{\mathrm{eq}}} f^{\mathrm{eq}}(\mathbf{Q}, \mathbf{P}) \tag{A.8}$$

where

$$f^{\mathrm{eq}}(\mathbf{Q}, \mathbf{P}) = \frac{1}{Z} \exp(-H(\mathbf{Q}, \mathbf{P})/k_B T), \tag{A.9}$$

$$P_{\alpha}^{\mathrm{eq}} = \frac{1}{h^{3N}} \int_{\Re_{\alpha}^{3N}} d^{3N}\mathbf{Q} \int d^{3N}\mathbf{P} f^{\mathrm{eq}}(\mathbf{Q}, \mathbf{P}), \tag{A.10}$$

$$\begin{aligned} Z &= \frac{1}{h^{3N}} \int d^{3N}\mathbf{Q} \int d^{3N}\mathbf{P} \exp(-H(\mathbf{Q}, \mathbf{P})/k_B T) \\ &= \sum_{\alpha} \frac{1}{h^{3N}} \int_{\Re_{\alpha}^{3N}} d^{3N}\mathbf{Q} \int d^{3N}\mathbf{P} \exp(-H(\mathbf{Q}, \mathbf{P})/k_B T) = \sum_{\alpha} Z_{\alpha}. \end{aligned} \tag{A.11}$$

In Eqs. (A.9) and (A.11), k_B is the Boltzmann constant and T is temperature. Under LEA, Eq. (A.6) is reduced to the master equation for discrete state transitions:

$$\frac{dP_\alpha(t)}{dt} = -\sum_\beta W_{\beta\alpha}P_\alpha(t) + \sum_\beta W_{\alpha\beta}P_\beta(t), \tag{A.12}$$

where the interstate transition probabilities are given by

$$W_{\beta\alpha} = \frac{1}{Z_\beta}\int d^{3N}\mathbf{P}\int_{\partial\Re^{3N}_{\beta\alpha}} d\mathbf{s}\bullet\mathbf{M}^{-1}\bullet\mathbf{P}\Theta\left(d\mathbf{s}\bullet\mathbf{M}^{-1}\bullet\mathbf{P}\right) \times \exp(-H(\mathbf{Q},\mathbf{P})/k_BT). \tag{A.13}$$

The surface integration in Eq. (A.13) is dominated by the contribution from the saddle point $\mathbf{Q}^{\text{ts}}_{\beta\alpha} \in \partial\Re^{3N}_{\beta\alpha}$ (*i.e.*, where the energy is minimum among all surface points), at which $\partial V/\partial\mathbf{Q}^{\text{ts}}_{\beta\alpha} = 0$ and all the eigenvalues of the Hessian matrix, $\partial^2V/\partial\mathbf{Q}^2|_{\mathbf{Q}=\mathbf{Q}^{\text{ts}}_{\beta\alpha}}$, are positive except for the one corresponding to the eigenvector that is normal to $\partial\Re^{3N}_{\beta\alpha}$. Let q_1 be the reaction coordinate along the Hessian eigenvector with the negative eigenvalue and p_1 be the corresponding momentum. The transition probability in Eq. (A.13) then becomes

$$W_{\beta\alpha} = \frac{k_BT}{h}\frac{Z^{\text{ts}}_{\beta\alpha}}{Z_\alpha} \tag{A.14}$$

where

$$Z^{\text{ts}}_{\beta\alpha} = \frac{1}{h^{3N-1}}\int d^{3N-1}\mathbf{Q} \times \int d^{3N-1}\mathbf{P}[\exp(-H(\mathbf{Q},\mathbf{P})/k_BT)]_{q_1=p_1=0}. \tag{A.15}$$

In harmonic TST, Eq. (A.15) is further simplified as

$$W_{\beta\alpha} = \frac{1}{2\pi}\exp\left(-\frac{\Delta_{\beta\alpha}}{k_BT}\right)\frac{\prod_{j=1}^{3N}\omega_j^\alpha}{\prod_{j=2}^{3N}\omega_j^{\beta\alpha}}, \tag{A.16}$$

where $\Delta_{\beta\alpha} = V(\mathbf{Q}^{\text{ts}}_{\beta\alpha}) - V(\mathbf{Q}^{\text{min}}_\alpha)$ is the activation barrier, ω_j^α and $\omega_j^{\beta\alpha}$ are the eigenvalues of the dynamical matrix, $\mathbf{D}_{ij} = (m_{\text{mod}(i-1,3)+1}m_{\text{mod}(j-1,3)+1})^{-1/2}\partial^2V/\partial\mathbf{Q}_i\partial\mathbf{Q}_j \in \Re^{3N\times3N}$, at $\mathbf{Q}^{\text{min}}_\alpha$ and $\mathbf{Q}^{\text{ts}}_{\beta\alpha}$, respectively (we define $\omega_1^{\beta\alpha}$ to be the negative eigenvalue).

The KMC method [15–20] simulates discrete state transitions governed by the master equation, Eq. (A.12). Suppose that the system is in state α at time $t = 0$. The probability, $P(t)dt$, with which the next event occurs in the small time interval, $[t, t + dt]$, is determined as follows. In a Poisson process, the probability of an event, $\alpha \to \beta$, to occur during dt is given by $W_{\beta\alpha}dt$ independent of the history. The probability that no event occurs in $[0, t]$ and one of the events occurs in $[t, t + dt]$ is given by

$$P(t)dt = \lim_{dt\to0}(1 - Wdt)^{t/dt}Wdt = W\exp(-Wt)dt, \tag{A.17}$$

where $W = \sum_\beta W_{\beta\alpha}$ is the sum of all possible events starting from state α. Let ξ_1 be a uniform random number in [0,1]. Then the random variable,

$$t = -\ln(\xi_1)/W, \tag{A.18}$$

follows the Poisson distribution, Eq. (A.17). KMC simulation consists of a time-stepping loop, where the time is incremented by Eq. (A.18) at each time step. The next state β is stochastically chosen with the probability, $P_\beta = W_\beta/W$. This is achieved by generating a second uniform random number ξ_2 and selecting β such that

$$\beta = \min_c\left\{\sum_{b=1}^c W_{b\alpha} > W\xi_2\right\}. \tag{A.19}$$

Appendix B. Derivation of the synchronous parallel kinetic Monte Carlo (SPKMC) algorithm

Scalable stochastic simulation of the master equation, Eq. (A.12), is possible, based on an observation that in most activated events, the particle displacements from a local energy minimum configuration $\mathbf{Q}^{\text{min}}_\alpha$ via a transition state $\mathbf{Q}^{\text{ts}}_{\beta\alpha}$ to a new local energy minimum $\mathbf{Q}^{\text{min}}_\beta$ are localized in space [23]. In such a case, we partition the 3-dimensional space $\Re^3$ into spatially localized domains,

$$\Re^3 = \bigcup_d \Re^3_d; \quad \Re^3_d \cap \Re^3_{d'} = \varnothing. \tag{B.1}$$

We assume that the domain size is larger than the cutoff distance beyond which events are statistically independent. At a given local minimum-energy configuration, we accordingly partition the set of N particles into subsets, so that there are N_d particles in domain $\Re^3_d$ ($N = \sum_{d=1}^D N_d$, where D is the number of domains). We denote the positions of the N_d particles by $\mathbf{Q}_d \in \Re^{3N_d}$.

In view of the spatial locality of events, we introduce a local transition-state (LTS) approximation as follows. For a given local minimum-energy configuration $\mathbf{Q}^{\text{min}}_\alpha$, the connected neighbor states $\mathbf{Q}^{\text{min}}_\beta$ can be enumerated by: (i) first specifying a spatial domain d in which an event that connects the two states via $\mathbf{Q}^{\text{ts}}_{\beta\alpha}$ occurs (*i.e.*, the differences between $\mathbf{Q}^{\text{min}}_\alpha$, $\mathbf{Q}^{\text{ts}}_{\beta\alpha}$ and $\mathbf{Q}^{\text{min}}_\beta$ are negligible outside $\Re^3_d$); and (ii) then enumerating distinct events β' that occur in domain d. In other words, an event is indexed as a pair,

$$\textbf{\textit{LTS approximation}}: \beta = \left(d, \beta'\right). \tag{B.2}$$

We then adopt the time-dependent Hartree (TDH) approximation to the Liouville equation [29], in which the probability density function is factored as

$$\textbf{\textit{TDH approximation}}: f(\mathbf{Q},\mathbf{P},t) \cong \prod_d f_d\left(\mathbf{Q}_d,\mathbf{P}_d,t\right), \tag{B.3}$$

i.e., local events are statistically independent and may be sampled independently of those in the other domains. In particular, the calculation of the transition probability in Eq. (A.16) is performed locally in each domain $\Re^3_d$.

We enumerate multiple local events at different domains $\Re^3_d$; let $N^{(d)}_{\text{evt}}$ be the number of events in domain d. Here, at most one event occurs at each domain d, and thus the set of events β in Eq. (B.2) is replaced by $\{e|e = 1,\ldots,N^{(d)}_{\text{evt}}\}$, which is a subset of the set of D domains. For the current state α in domain d, let

$$W^{(d)} = \sum_\beta W_{(d',\beta)(d,\alpha)} \tag{B.4}$$

be the sum of all possible next states. Here, we assume the locality of state transitions such that the consecutive state positions are bounded by a cutoff radius q_{cut}. While the domain d', to which the destination state β resides, may or may not be the original domain d, the location of β is within radius q_{cut} from the d–d' boundary. We next define

$$W_{\text{max}} = \max_d\left\{W^{(d)}\right\}, \tag{B.5}$$

and assign a null event (*i.e.*, the state remains the initial α state) with the rate

$$W^{(d)}_0 = W_{\text{max}} - W^{(d)}, \tag{B.6}$$

following the synchronous parallel KMC algorithm [25–28].

Under the LTS and TDH approximations, KMC simulation consists of a time-stepping loop, where the time at each KMC step is incremented by

$$t = -\ln(\xi_1)/W_{\text{max}}, \tag{B.7}$$

where ξ_1 is a uniform random number in the range [0,1]. Within each domain d, the next state β is stochastically chosen with the probability, $P_\beta^{(d)} = W_{(d',\beta)(d,\alpha)}/W_{\max}$. This is achieved by generating a second uniform random number $\xi_2^{(d)}$ per domain and selecting β such that

$$\beta = \min_c \left\{ \sum_{b=1}^{c} W_{(d',b)(d,\alpha)} > W_{\max}\xi_2^{(d)} \right\}. \tag{B.8}$$

References

[1] D.A. Reed, J. Dongarra, Commun. ACM 58 (2015) 56–68.
[2] N.A. Romero, A. Nakano, K. Riley, F. Shimojo, R.K. Kalia, P. Vashishta, P.C. Messina, IEEE Computer 48 (11) (2015) 33–41.
[3] L. Greengard, V. Rokhlin, J. Comput. Phys. 73 (1987) 325–348.
[4] K. Nomura, H. Dursun, R. Seymour, W. Wang, R.K. Kalia, A. Nakano, P. Vashishta, F. Shimojo, L.H. Yang, Proc. Int. Parallel Distrib. Process Symp., IPDPS 2009, 2009, IEEE.
[5] D.R. Bowler, T. Miyazaki, Rep. Progr. Phys. 75 (2012) 036503.
[6] F. Shimojo, R.K. Kalia, M. Kunaseth, A. Nakano, K. Nomura, S. Ohmura, K. Shimamura, P. Vashishta, J. Chem. Phys. 140 (2014) 18A529.
[7] D.E. Shaw, R.O. Dror, J.K. Salmon, J.P. Grossman, K.M. Mackenzie, J.A. Bank, C. Young, M.M. Deneroff, B. Batson, K.J. Bowers, E. Chow, M.P. Eastwood, D.J. Ierardi, J.L. Klepeis, J.S. Kuskin, R.H. Larson, K. Lindorff-Larsen, P. Maragakis, M.A. Moraes, S. Piana, Y. Shan, B. Towles, Proc. Supercomputing, SC09, 2009, IEEE/ACM.
[8] D. Perez, B.P. Uberuaga, Y. Shim, J.G. Amar, A.F. Voter, Ann. Rep. Comput. Chem. 5 (2009) 79–98.
[9] D.G. Truhlar, B.C. Garrett, S.J. Klippenstein, J. Phys. Chem. 100 (1996) 12771–12800.
[10] P. Hanggi, P. Talkner, M. Borkovec, Rev. Modern Phys. 62 (1990) 251–341.
[11] A.F. Voter, Phys. Rev. B 57 (1998) R13985–R13988.
[12] A. Nakano, Comput. Phys. Comm. 176 (2007) 292–299.
[13] A. Nakano, Comput. Phys. Comm. 178 (2008) 280–289.
[14] K.J. Kohlhoff, D. Shukla, M. Lawrenz, G.R. Bowman, D.E. Konerding, D. Belov, R.B. Altman, V.S. Pande, Nature Chem. 6 (2014) 15–21.
[15] A.B. Bortz, M.H. Kalos, J.L. Lebowitz, J. Comput. Phys. 17 (1975) 10–18.
[16] D.T. Gillespie, J. Comput. Phys. 22 (1976) 403–434.
[17] K.A. Fichthorn, W.H. Weinberg, J. Chem. Phys. 95 (1991) 1090–1096.
[18] A.F. Voter, in: K.E. Sickafus, E.A. Kotomin, B.P. Uberuaga (Eds.), Radiation Effects in Solids, Springer, Dordrecht, The Netherlands, 2006.
[19] A.P.J. Jansen, An Introduction To Kinetic Monte Carlo Simulations of Surface Reactions, Springer, Berlin, Germany, 2012.
[20] W. Mou, S. Hattori, P. Rajak, F. Shimojo, A. Nakano, Appl. Phys. Lett. 102 (2013) 173301.
[21] J.L. Blue, I. Beichl, F. Sullivan, Phys. Rev. E 51 (1995) R867–R868.
[22] T.P. Schulze, Phys. Rev. E 65 (2002) 036704.
[23] G.T. Barkema, N. Mousseau, Phys. Rev. Lett. 81 (1998) 1865–1868.
[24] G. Korniss, M.A. Novotny, H. Guclu, Z. Toroczkai, P.A. Rikvold, Science 299 (2003) 677–679.
[25] Y. Shim, J.G. Amar, Phys. Rev. B 71 (2005) 115436.
[26] Y. Shim, J.G. Amar, Phys. Rev. B 71 (2005) 125432.
[27] E. Martinez, J. Marian, M.H. Kalos, J.M. Perlado, J. Comput. Phys. 227 (2008) 3804–3823.
[28] E. Martinez, P.R. Monasterio, J. Marian, J. Comput. Phys. 230 (2011) 1359–1369.
[29] R. Elber, M. Karplus, J. Am. Chem. Soc. 112 (1990) 9161–9175.
[30] H.S. Byun, S. Pirbadian, A. Nakano, L. Shi, M.Y. El-Naggar, ChemElectroChem 1 (2014) 1932–1939.
[31] C.M. Nakano, H.S. Byun, H. Ma, T. Wei, M.Y. El-Naggar, Comput. Phys. Comm. 193 (2015) 1–9.
[32] F.H. Stillinger, Phys. Rev. E 59 (1999) 48–51.
[33] M.P. Allen, D.J. Tildesley, Computer Simulation of Liquids, Oxford University Press, New York, 1987.
[34] A. Nakano, R.K. Kalia, P. Vashishta, Comput. Phy. Comm. 83 (1994) 197–214.
[35] A. Nakano, Concurrency, Pract. Exp. 11 (1999) 343–353.
[36] W. Gropp, E. Lusk, A. Skjellum, Using MPI, third ed., MIT Press, Cambridge, MA, 2014.
[37] D.J. Richardson, J.N. Butt, J.K. Fredrickson, J.M. Zachara, L. Shi, M.J. Edwards, G. White, N. Baiden, A.J. Gates, S.J. Marritt, T.A. Clarke, Mol. Microbiol. 85 (2012) 201–212.
[38] M.Y. El-Naggar, G. Wanger, K.M. Leung, T.D. Yuzvinsky, G. Southam, J. Yang, W.M. Lau, K.H. Nealson, Y.A. Gorby, Proc. Natl. Acad. Sci. USA 107 (2010) 18127–18131.
[39] S. Pirbadian, S.E. Barchinger, K.M. Leung, H.S. Byun, Y. Jangir, R.A. Bouhenni, S.B. Reed, M.F. Romine, D.A. Saffarini, L. Shi, Y.A. Gorby, J.H. Golbeck, M.Y. El-Naggar, Proc. Natl. Acad. Sci. USA 111 (2014) 12883–12888.
[40] R.M. Snider, S.M. Strycharz-Glaven, S.D. Tsoi, J.S. Erickson, L.M. Tender, Proc. Natl. Acad. Sci. USA 109 (2012) 15467–15472.
[41] M.D. Yates, J.P. Golden, J. Roy, S.M. Strycharz-Glaven, S. Tsoi, J.S. Erickson, M.Y. El-Naggar, S.C. Barton, L.M. Tender, Phys. Chem. Chem. Phys. 17 (2015) 32564–32570.
[42] G.F. White, Z. Shi, L. Shi, Z.M. Wang, A.C. Dohnalkova, M.J. Marshall, J.K. Fredrickson, J.M. Zachara, J.N. Butt, D.J. Richardson, T.A. Clarke, Proc. Natl. Acad. Sci. USA 110 (2013) 6346–6351.
[43] B.J. Gross, M.Y. El-Naggar, Rev. Sci. Instrum. 86 (2015).
[44] K. Andersen, S. Latini, K.S. Thygesen, Nano Lett. 15 (2015) 4616–4621.
[45] A. Nakano, S. Ichimaru, Phys. Rev. B 39 (1989) 4930–4937.
[46] R. Zwanzig, Nonequilibrium Statistical Mechanics, Oxford University Press, Oxford, UK, 2001.