Synchronous relaxation algorithm for parallel kinetic Monte Carlo simulations of thin film growth

Michael Merrick and Kristen A. Fichthorn*
Departments of Chemical Engineering and Physics, The Pennsylvania State University, University Park, Pennsylvania 16802, USA
(Received 14 July 2006; published 30 January 2007)

We present an optimistic synchronous relaxation algorithm for parallel kinetic Monte Carlo (KMC) simulations of thin film growth. This algorithm is based on spatial decomposition of the KMC lattice and it employs two measures aimed at improving the parallel efficiency: dynamic global updating and domain boundary shifting. We utilize this algorithm to simulate two different growth models, which represent the growth of Ag on Ag(111) and the heteroepitaxial growth of Ag on Pt(111). We show that these simulations can achieve good efficiency—especially for large domain sizes with a moderate number of processors. We find that domain boundary shifting can improve efficiency—especially for simulations of growth in the Ag/Pt(111) system, where the potential-energy surface topology creates areas of rapid, localized motion. We analyze the origins of parallel efficiency in these simulations.

DOI: 10.1103/PhysRevE.75.011606 PACS number(s): 81.15.Aa, 05.10.Ln, 89.20.Ff

I. INTRODUCTION

Kinetic Monte Carlo (KMC) simulations have emerged as an attractive computational tool for modeling systems whose dynamical evolution is governed by rare events. Although these simulations have almost exclusively been done on serial computers, KMC simulations can be run on parallel architectures and reap benefits in speedup and in the system size that can be probed. In this work, we study a parallel implementation of KMC for modeling submonolayer epitaxial growth.

We focus on the "optimistic" synchronous relaxation (SR) algorithm, which was originally developed by Eick *et al.* [1] to model phone networks, employed by Lubachevsky and Weiss for Ising spin systems [2], and adapted for epitaxial growth simulations by Shim and Amar [3]. As we will discuss below, the SR algorithm is based on spatial decomposition of the KMC simulation lattice and, in maintaining the fidelity of the simulated KMC timeline, it can incur significant parallel overhead. In this work, we apply the SR algorithm to simulate the growth of Ag on Ag(111) and of Ag on one monolayer (ML) of Ag on Pt(111) [4–7]. The Ag/Ag(111) system is similar to the systems studied by Shim and Amar [3], in that adsorbed atoms have nearest-neighbor interactions—although Shim and Amar studied growth on a square lattice instead of the fcc(111) lattice of fcc and hcp sites that we study here. For simulations of the growth of Ag on 1-ML Ag on Pt(111), we include interactions, based on calculations with density-functional theory (DFT), up to the sixth neighbor [8]. We demonstrate that the SR algorithm can provide good efficiency in parallel simulations for both these systems, although the overhead can be substantial. By incorporating boundary shifting into the SR algorithm, additional efficiency can be gained. We quantify the origins of efficiency in these simulations.

*Email address: fichthorn@psu.edu

II. MODELS AND METHODS

A. Growth models

We simulate the submonolayer growth of Ag on Ag(111) and of Ag on 1-ML Ag on Pt(111). This involves simulating deposition with a rate of F onto open fcc or hcp sites and adatom hopping between fcc and hcp binding sites on these fcc(111) surfaces. We note that only "open" sites which (i) are not occupied; and (ii) have the three adjacent binding sites unoccupied can receive a deposited atom or have a neighboring atom hop into them. The hopping rate of a particular adatom from an initial site i to a final, open site f depends on the neighboring adatoms to sites i and f. This dependence can be understood via the framework of transition-state theory (TST), which predicts a rate $r_{i\to f}$ of the form

$$r_{i\to f} = \nu_{0,i\to f} \exp\left(\frac{-E_{i\to f}}{kT}\right), \tag{1}$$

where $E_{i\to f}$ is the energy barrier to go from site i to f, k is Boltzmann's constant, T is temperature, and $\nu_{0,i\to f}$ is the prefactor. In our simulations, the energy barrier is given by [9]

$$E_{i\to f} = E^0 + \frac{1}{2}(\epsilon_f - \epsilon_i), \tag{2}$$

where E^0 is the energy barrier for hopping of an isolated adatom and $\epsilon_{i(f)}$ represents the sum of pairwise interaction energies at the initial (final) state. We utilize a constant prefactor that is independent of the local environment of the adatoms, so $\nu_{0,i\to f} = \nu_0 = 10^{12}$ s^{-1}.

For the Ag on Ag(111) system, we used $E^0 = 87$ meV and a nearest-neighbor pair interaction of 190 meV—both values were obtained from first-principles DFT calculations [4]. For Ag on 1-ML Ag/Pt(111), we used $E^0 = 52$ meV [4]. The values of the pair interaction energies for the first through sixth neighbors used to model this system are summarized in Table I and their spatial layout is shown in the diagram of Fig. 1. These interactions are a subset of those obtained in the DFT

 ©2007 The American Physical Society

TABLE I. Pair interactions for Ag on 1-ML Ag on Pt(111).

Neighbor rank	Interaction (meV)
1	−190
2	−138
3	30
4	50
5	30
6	20

study by Luo and Fichthorn [8], who quantified these interactions up to the 53rd neighbor. We note that the three "zeroth neighbor sites" to an adatom remain vacant with fcc packing. Also the DFT calculations indicate a difference in binding energy between fcc and hcp sites of less than 3 meV and that the pair interaction is independent of whether two atoms are on fcc or hcp binding sites [4].

With diffusion barriers of the form given by Eq. (2), there are high energy barriers for atoms in islands to rearrange once they have aggregated. This would lead to fractal adsorbate islands, shaped like diffusion-limited aggregates (see, for example, Ref. [10]) with only a nearest-neighbor attraction. In the presence of the pair interactions given in Table I, limited island rearrangement leads to elongated, chainlike islands [5,6]. As discussed elsewhere [5], the kinetics of island rearrangement cannot be adequately modeled by nearest-neighbor hops and likely occurs via complex, multiatom motions. To account for these processes, we allow an fcc (hcp) atom with one or more nearest neighbors to hop to a nearest-neighbor fcc (hcp) site, thereby avoiding the intermediate hcp (fcc) site. These hops occur in addition to hops between neighboring fcc and hcp sites. The energy barriers for these longer hops are of the form given by Eq. (2) with E^0 set to a value 1.5 times higher than the hopping barrier for an isolated atom. This value was chosen to allow for island rearrangement without imparting significant island mobility. In this way, we achieve compact islands similar to those seen in experimental studies at these conditions [11,12].

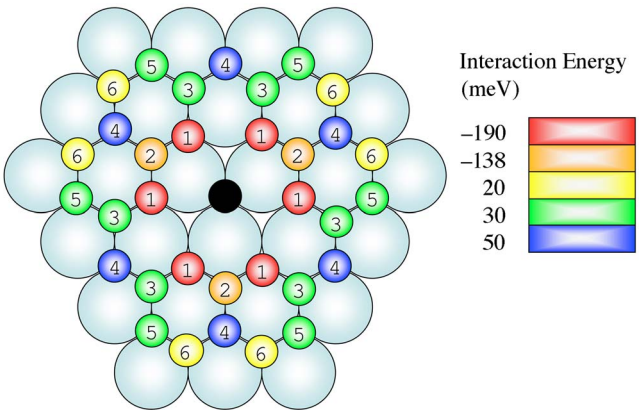

FIG. 1. (Color online) Diagram indicating the spatial layout of interactions for the Ag/1-ML Ag/Pt(111) system. The positions of the first through sixth neighbors are indicated and the scale (also Table I) indicates the strength of the interaction at each distance.

We probed submonolayer growth in both these systems. When atoms are deposited onto an initially bare substrate, they diffuse and aggregate to form nuclei, which grow to stable islands with the addition of diffusing adatoms. In the initial stages of growth, the number of islands (or their density) increases as new islands nucleate. After the island density has reached a steady-state value, the probability that newly deposited adatoms will add to existing islands exceeds the probability that they will nucleate new islands. In this steady-state regime, the number of islands is approximately constant until islands grow large enough that they coalesce and the island density decreases. In our work, we simulate growth beginning from an initially bare substrate up to the "steady-state" regime, at a coverage in the range of 0.1–0.2 ML. For both systems, we simulated the temperature range from 35 to 65 K, with a deposition rate of $F=0.1$ ML/s. An important ratio characterizing growth is F/D, or the ratio of the deposition rate F to the rate for isolated adatom hopping D. For the conditions probed here, we cover the range $F/D=5.6\times10^{-7}$–$F/D=0.34$ for Ag/Ag(111) and $F/D=1.1\times10^{-9}$–$F/D=3.1\times10^{-6}$ for Ag/1-ML Ag/Pt(111).

B. Serial KMC algorithm

We use a serial KMC algorithm based on the general KMC method proposed by Fichthorn and Weinberg [13], which adapts the "N-fold way" algorithm of Bortz, Kalos, and Liebowitz [14]. For a lattice consisting of N sites, the algorithm proceeds as follows:

(i) Determine the total rate $r_{T,i}$ associated with each site i. The total rate is the sum of the rates r_j of all diffusion hops j originating from site i (i.e., $r_{T,i}=\Sigma r_j$), if site i is occupied, or the deposition rate F (i.e., $r_{T,i}=F$), if site i is vacant. While this algorithm would not be adequate for multilayer growth, it is sufficient to simulate the low coverages probed here.

(ii) Determine the total rate R_T for the global lattice, where

$$R_T=\sum_{i=1}^{N} r_{T,i}. \tag{3}$$

(iii) Select a site i at random, with a probability given by

$$P_i=\frac{r_{T,i}}{R_T}. \tag{4}$$

(iv) At site i, randomly select a particular transformation j with rate r_j and a probability given by

$$P_j=\frac{r_j}{r_{T,i}}. \tag{5}$$

(v) Actuate the event and increment the simulated time using

$$\Delta t=\frac{-\ln\rho}{R_T}, \tag{6}$$

where ρ is a uniform random number between 0 and 1.

(vi) Repeat the algorithm until a simulation criterion is met for exiting the KMC loop.

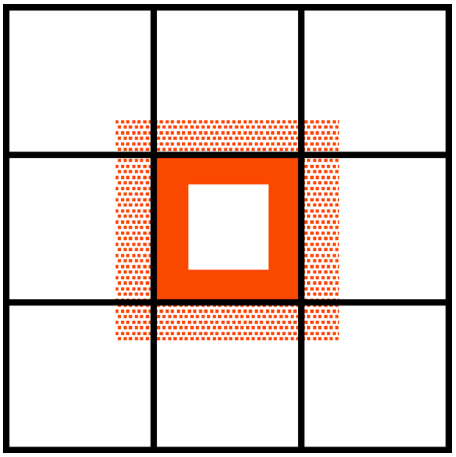

FIG. 2. (Color online) Domain decomposition of a KMC lattice into nine domains. The ghost region for one of the domains is indicated by the shaded area, while the skin region is indicated by the filled area, and the inner domain core is the central area in white.

C. Parallel SR-KMC algorithm

The parallel SR-KMC algorithm that we use here bears close similarity to that employed by Shim and Amar [3], although there are differences that we will discuss below. This algorithm is based on spatial decomposition of a large KMC lattice into M square domains, as shown in Fig. 2. If each domain was independent of the rest, a parallel KMC simulation would involve running the serial algorithm for each domain on a different processing element (PE) and combining the time lines of all domains to obtain the evolution of the global system. However, the domains are not independent of one another, as atoms in a given domain may interact with atoms in neighboring domains. Moreover, atoms may diffuse from one domain to another. Both of these features serve to couple timelines of the various domains. The bulk of the overhead involved in the SR-KMC algorithm is focused on properly handling domain interactions.

To account for domain interactions, each domain contains a core, a "skin" region, and a "ghost" region. As shown in Fig. 2, the core is the innermost area of a domain. Events that occur within cores are independent of events that occur on other domains. The skin sites occupy the region between the core and the domain boundary. These sites represent an area where a configurational change would affect rates in a neighboring domain. Thus the "thickness" of the skin region is defined by the interaction range among the adatoms. The skin thicknesses for the Ag/Ag(111) system and the Ag/1-ML Ag/Pt(111) system are one and four lattice sites, respectively. The ghost sites contain necessary information from neighboring domains to determine rates in the skin region. These sites are defined such that the ghost sites of a given domain are the skin sites of the neighboring domains, as shown in Fig. 2.

Using the domain decomposition described above, the parallel SR-KMC algorithm proceeds as follows:

(i) Each PE carries out a single, serial KMC iteration (as described above) on its prescribed domain.

(ii) Each PE ascertains if the chosen event occurred in its skin region. If the event on a particular PE is a "skin event," then this PE advances to step iv; otherwise the PE advances to step iii.

(iii) If the number of KMC iterations on a PE is less than the global update frequency G, then this PE goes back to step i; otherwise the PE advances to step iv.

(iv) The various PEs communicate with one another in a global update, which occurs when all PEs have reached this step. Each PE communicates to all the others the time of its last event and whether or not the event was a skin event.

(v) The least advanced time is determined on each PE as the smallest last event time among all PEs.

(vi) If necessary, each PE performs a SR iteration, i.e., it removes all the KMC events that occurred after the least advanced time. If a skin event has the least advanced time, then the neighboring PE(s) to the PE that hosted the skin event are updated to account for the new lattice configuration.

(vii) The domain boundaries and G, are adjusted, if necessary.

(viii) The algorithm recommences at step i, with each PE at the end of its timeline from the previous cycle, until a criterion is met for exiting the SR-KMC loop.

Thus the parallel SR-KMC algorithm consists of a cycle in which tentative, parallel, independent KMC timelines are first generated, then verified for consistency with one another, and finally revised via the SR step to ensure fidelity of the parallel simulation to an analogous serial simulation.

There are several notable features of our KMC algorithms. First, in our serial KMC algorithm, we choose events by selecting first an appropriate lattice site and then choosing an event from among those possible at that site. An alternate way of selecting events is to utilize event lists [14]. If the number of different possible events is small and can be determined before the simulation is executed, then utilization of event lists is efficient. However, in some of the newer implementations of KMC, where events are determined on-the-fly [17–19], it is not possible to construct event lists. In this work, the number of different possible events for the Ag/1-ML Ag/Pt(111) system is larger than the number of sites on our domains. In this case, it is more efficient to choose events by sites.

Choosing events by sites leads to a different scaling of our serial KMC algorithm with lattice size than that obtained by choosing events by lists. For example, the computational effort associated with Shim and Amar's algorithm [3], in which events are selected by lists, scales linearly with the lattice size N. However, our algorithm scales as $\sim N^2$: The number of iterations to achieve, for example, a fixed fractional adsorbate coverage scales as N and the effort per iteration associated with obtaining the total rate and site probabilities (i.e., steps ii and iii in the serial KMC algorithm) also scales as N. We have reduced the scaling of event selection (step iii of the serial KMC algorithm) by using a binary tree search [15], which scales as $\ln N$. It is also possible to reduce the scaling for obtaining the total rate (step ii) to $\ln N$ using a binary tree [16]. By both selecting events and calculating the total rate using binary trees, it is possible to have $N \ln N$

scaling of a site-based algorithm, but this is not implemented here.

An advantage of choosing events by sites is that the SR rollback algorithm is relatively straightforward to implement. Thus, here, we retain events from partially completed cycles, as described in step vi of the parallel KMC algorithm. When events are chosen from event lists, partial rollbacks are difficult. Thus, in Shim and Amar's study, an entire cycle is discarded if a skin event occurs [3].

In the absence of skin events, the KMC timelines on the various domains would be independent and the SR-KMC algorithm could achieve 100% (or even superparallel) efficiency. A substantial portion of parallel overhead is involved with detecting and properly dealing with skin events. Step vii in the above outline of the SR-KMC algorithm contains features to improve the efficiency of the SR-KMC algorithm in this regard. We discuss each of these features below.

The SR-KMC algorithm contains a global update frequency G, which is the maximum number of KMC iterations that can occur on a given PE before a global update occurs. Here, we consider that there is an optimal value for G, which reflects a balance between communication and computation overhead. Communication between PEs entails a significant amount of parallel overhead. Moreover, before a skin event occurs, the timelines on the various PEs are independent of one another. Thus, to reduce communication overhead, it is desirable not to communicate (possible) skin events after every serial KMC iteration. On the other hand, if G is larger than the frequency with which skin events occur, then computational overhead is consumed in "rollbacks" via the SR algorithm. As discussed by Shim and Amar [3], optimal efficiency can depend on several factors, including a third type of parallel overhead which occurs from fluctuations and associated PE idleness. They tried a number of strategies to determine the optimal global update frequency and found that and that dynamical optimization to attain some fixed quantity during a cycle was the best method.

To attempt to increase efficiency here, we dynamically alter the value the value of G to track the frequency of skin events observed in our simulations. We expect the skin-event frequency to change with time during our simulations, as the film morphology evolves. We begin with a base value of G_0 (=10), which was determined to be near optimal in short-time simulations of Ag/Ag(111) and Ag/1-ML Ag/Pt(111) with a fixed global update frequency. If all iterations can be completed without a skin event, we increment G by δG (=3) and we continue to increment G by δG for each cycle that is completed without a skin event. The value of $\delta G=3$ was determined to be optimal in simulations using $\delta G=1$, 2, 3, 5, and 10. When a skin event occurs, G is reset to its value of G_0 unless the skin event occurs before G_0 iterations, then we decrement the value of G below G_0. This feature adds a small amount of parallel overhead in the form of computation.

We also allow the domain boundaries to shift in our simulations. Occasionally, repeated skin events occur due to recurrence of a particular move sequence. The repetitious skin events lead to a loss of efficiency. If the domains can be shifted such that these events occur within the domain core regions, then efficiency can be gained. Thus, in monitoring the frequency of skin events, if we detect two consecutive skin events we shift the domain boundaries either left-right or up-down, depending on the location of the last skin event, as depicted in Fig. 3. The boundaries are uniformly shifted by the width of the skin region and periodic boundary conditions are maintained during the shift. The algorithm requires one additional global communication to transmit the locations of the atoms beyond the locations of the ghost sites of a particular domain.

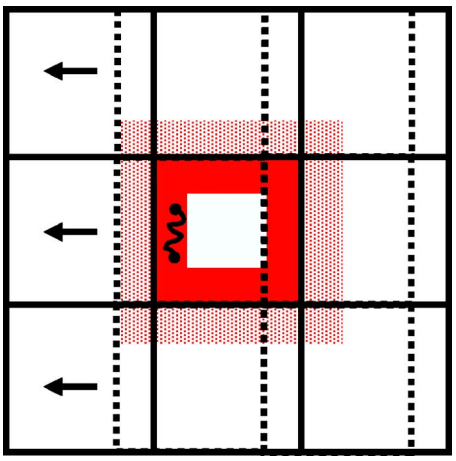

FIG. 3. (Color online) Illustration of the boundary-shift algorithm for a KMC lattice decomposed into nine domains. Repeated motion in the skin region (indicated by the small trajectory) leads to frequent global updates and a loss of efficiency. Here, the original domain boundaries, given by solid lines, are shifted to the left and the new boundaries are indicated by dashed lines. In this way, the region hosting repetitive skin events is shifted into the core region.

The KMC simulations were carried out on three different computing platforms. The first of these (denoted as C1) consists of 20 AMD Athlon XP nodes with a clock speed of 1.6 GHz and a fast-ethernet communication network. The second (C2) is a Beowulf cluster consisting of 162 AMD Athlon MP2200+ nodes with a clock speed of 2.8 GHz and with high-speed Dolphin and fast-ethernet network interconnects. The third cluster (C3) consists of 128 Intel P4 nodes with a clock speed of 2.4 GHz and a Quadrics high-speed network interconnect. The codes were written in FORTRAN. All interprocessor communications were carried out using the Message-Passing Interface (MPI).

III. RESULTS AND DISCUSSION

We first validated the parallel code by comparing the simulated island density N_x between the serial and parallel codes. The island density is defined as the number of islands per lattice site and is shown in Fig. 4 as a function of lattice size for various temperatures and numbers of PEs on various platforms for the Ag on Ag(111) system. The parallel island densities agree with the serial values and depend only on temperature, as we expect if the parallel code is executing properly. Analogous results are seen in our parallel simulations of Ag on 1-ML Ag/Pt(111).

The performance of a parallel computer code is normally characterized by two quantities, the speedup S and the effi-

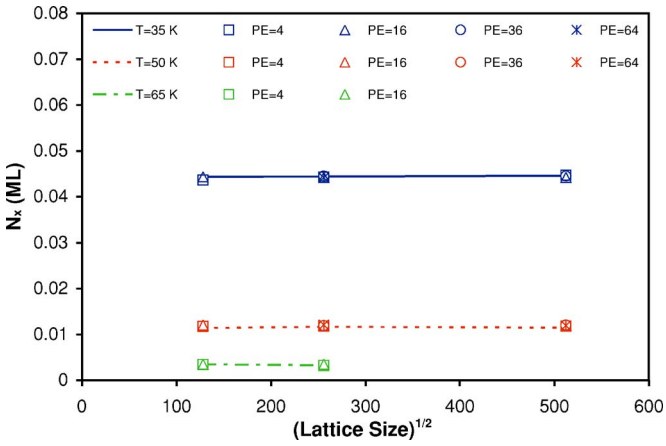

FIG. 4. (Color online) Simulated island density N_x as a function of lattice size for the Ag/Ag(111) system at T= 35, 50, and 65 K. The lines indicate averages obtained in serial simulations, while the symbols represent various parallel results. The uncertainties are the standard error and are smaller than the symbol sizes.

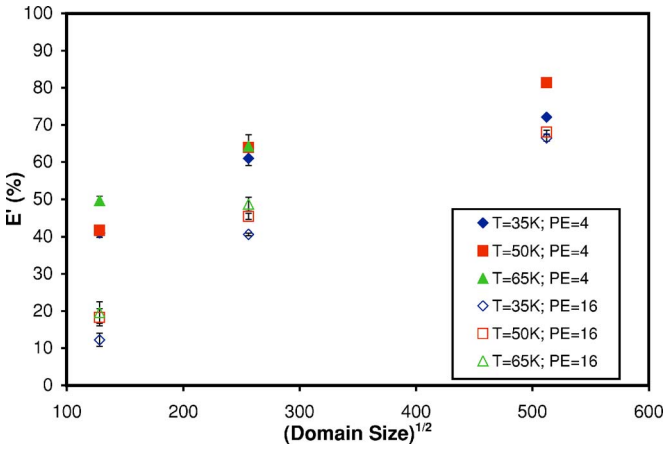

FIG. 5. (Color online) Parallel efficiency, as defined by Eq. (9), for simulations of the Ag/Ag(111) system on C1 as a function of the domain size for various temperatures with 4 and 16 PEs. The uncertainties are the standard error and are smaller than the symbol sizes when not shown.

ciency E. If t_s is the time required by a single PE to execute the serial code and t_p is the time required to execute the parallel code on P PEs, then standard definitions for S and E are

$$S = \frac{t_s}{t_p}, \tag{7}$$

and

$$E = \frac{S}{P} \times 100\,\% . \tag{8}$$

For a serial simulation that scales linearly with the problem size, the ideal parallel efficiency is 100% (i.e., the parallel code will run P times faster on P processors). Because of the specifics of the serial KMC algorithm employed here (as discussed above, our serial algorithm scales with the lattice size N as $\sim N^2$), our parallel codes exhibit superparallel efficiencies (i.e., efficiencies greater than 100%) using the definition of Eq. (8). Thus, to better quantify the additional parallel overhead incurred here, we adopt a different efficiency, defined by

$$E' = \frac{t_{s,N}}{t_{p,PN}} \times 100\,\% , \tag{9}$$

where $t_{s,N}$ is the time required to execute the serial code on a lattice containing N sites and $t_{p,PN}$ is the time required to run the parallel code on a lattice containing $P \times N$ sites distributed equally over P processors. If the parallel code is 100% efficient, then the time required to run the parallel code on P processors with domains of size N is equal to the time to run a serial simulation on a single lattice of size N, independent of the scaling of the code with N. Equation (9) provides a clear measure to quantify the overhead incurred by the parallel algorithm and this efficiency will be reported throughout the rest of the paper.

A. Ag on Ag(111)

Figure 5 shows the parallel efficiency as a function of the domain size for simulations of the Ag/Ag(111) system on C1. As might be anticipated, the efficiency increases with increasing domain size. In these simulations, efficiency is connected to the frequency of skin events, which decreases as the domain size increases and the ratio of skin to core sites decreases. We also see the effect of temperature, which is weak. At first, this might be considered surprising, since adatom hopping into the skin region comprises the majority of skin events. The adatom hopping rates increase exponentially with increasing temperature, leading to an increasing frequency of skin events and a loss of efficiency. However, a second loss of efficiency occurs at low temperatures, where PE idleness and "rollbacks" become important. At low temperatures, adatom mobilities are low and we can simulate a greater average number of events per cycle (i.e., G is larger at low temperatures). However, fluctuations in the simulated number of events per cycle, due to variations in the distribution of skin-event times, increase as the temperature decreases. The result is increasing PE idleness and overhead for SR rollbacks as temperature decreases. These opposing trends lead to a weak temperature dependence of our results.

Comparing results for 4 and 16 PEs in Fig. 5, we see that the efficiency for a fixed domain size is lower for 16 PEs than it is for 4 PEs. This trend is also seen in Fig. 6, which shows the parallel efficiencies found on C2 and C3 for the growth of Ag on Ag(111). We see that the efficiency is a decreasing function of the number of processors. Similar decreases have been observed elsewhere [2,3,20]. The decreasing efficiencies occur because of an increasing number of communications on global updates, as well as increased processor idleness (and its associated increase in SR rollbacks). Concerning the latter, a global update cannot occur until all processors have reached step iv in the above outline of the parallel algorithm. Fluctuations in the number of events per cycle increase with increasing number of PEs for systems

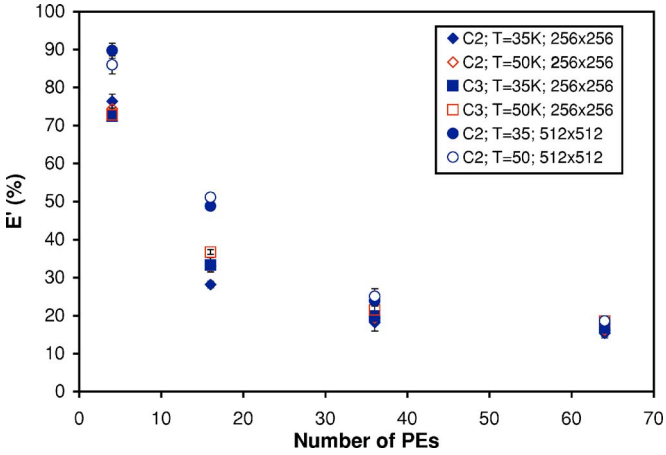

FIG. 6. (Color online) Parallel efficiency, as defined by Eq. (9), for simulations of the Ag/Ag(111) system on C2 and C3 as a function of the number of PEs for two different temperatures. The uncertainties are the standard error and are smaller than the symbol sizes when not shown.

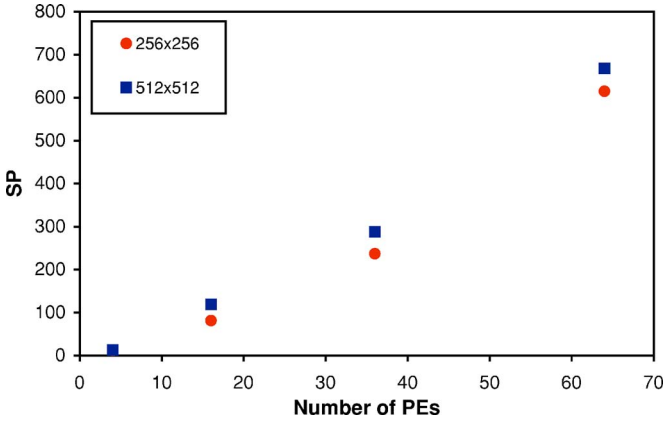

FIG. 7. (Color online) Speedup, as defined by Eq. (7), on C2 as a function of the number of PEs for simulations of Ag/Ag(111) at a temperature of 50 K for two different domain sizes.

that require a global update [2,3,20] and there is overhead in idleness and SR rollbacks associated with this. Comparing results from simulations with 256×256 and 512×512 domains in Fig. 6, we see that the larger domains have higher efficiencies, as we also saw in Fig. 5. However, the difference between the larger and smaller domains decreases with an increasing number of PEs. Again, we can attribute this to the effects of idleness due to fluctuations that occur with an increasing number of processors. For a large number of processors, these fluctuations begin to outweigh benefits in the increased core-to-skin ratio for a large domain size. Finally, comparing Figs. 5 and 6, we see that the efficiencies for a fixed temperature, domain size, and number of PEs are $\sim$10–15 % higher on C2 and C3 than they are on C1. This difference occurs because of the faster network communication speed and lower latency on C2 and C3 than that on the fast-ethernet network on C1. For example, we find that communication and PE idleness account for 98% of the parallel overhead in the Ag/Ag(111) simulations performed on C1 at a temperature of 50 K and with a domain size of 128 and 4 PEs.

It is worth mentioning that, although the parallel efficiencies can be low, significant speed-ups can be achieved via the SR-KMC algorithm. Figure 7 shows the speedup, obtained using Eq. (7), for the Ag/Ag(111) system. Using the definition of Eq. (8), these speedups would lead to superparallel (greater than 100%) efficiencies. It is important to note that the achievable speedup and efficiency, as defined by Eq. (8), depend on the implementation of steps i–iii in the serial KMC algorithm and their scaling with lattice size, as discussed above. By altering the lattice-size scaling of our serial KMC code, the efficiencies defined by Eqs. (8) and (9) would be comparable and the speedups would be more modest than those that we find.

B. Ag on 1-ML Ag/Pt(111)

The parallel efficiencies for the simulations of the growth of Ag on 1-ML Ag/Pt(111) are shown in Fig. 8. Although these efficiencies exhibit the same trends as those shown in Figs. 5 and 6, they are overall lower than those for the Ag/Ag(111) system. We attribute this to the larger skin region needed to capture the first- through sixth-neighbor interactions for the heteroepitaxial system. With a larger skin thickness, skin events are more frequent and the overhead associated with SR rollbacks is greater. For example, calculation is the most significant source of parallel overhead, comprising 74%, in simulations of Ag/1-ML Ag/Pt(111) on C2 for a 128×128 domain, with four PEs, at a temperature of 50 K. We note that the efficiency does increase with increasing domain size and, in attempts to increase efficiency, it should be possible to compensate for larger skin thicknesses by choosing larger domain sizes—as long as the number of processors remains moderate.

We note that the efficiency decreases as the number of processors increases in Fig. 8, as we saw in Figs. 5 and 6 for Ag/Ag(111). Shim and Amar noted a similar decrease in their studies [3]. They found that the dependence of efficiency on the number of processors (N_P) could be described by

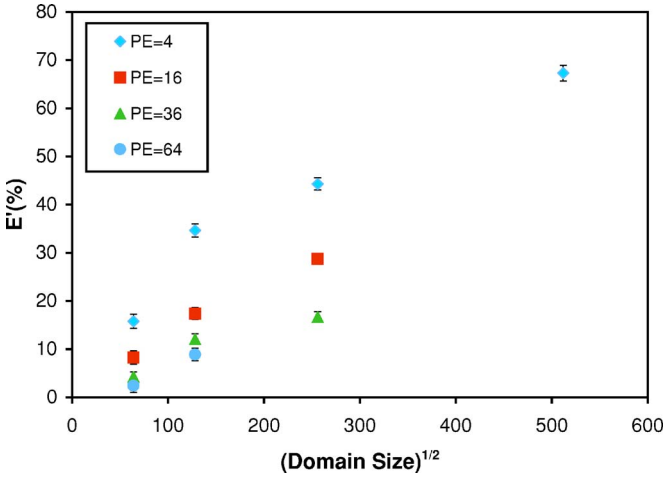

FIG. 8. (Color online) Parallel efficiency, as defined by Eq. (9), on C2 as a function of the domain size for simulations of the Ag/1-ML Ag/Pt(111) system at 35 K. The uncertainties are the standard error and are smaller than the symbol sizes when not shown.

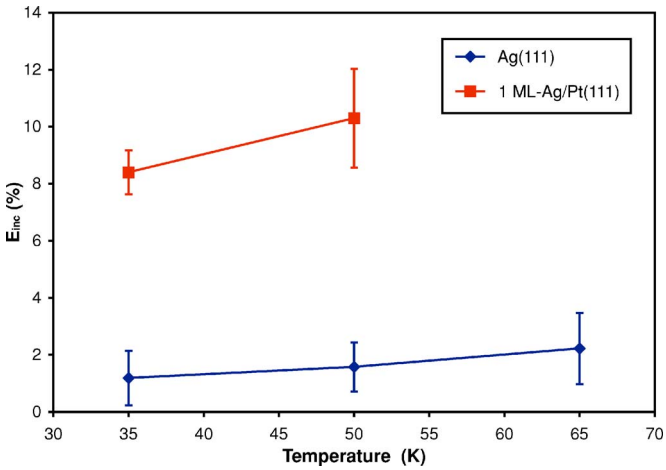

FIG. 9. (Color online) Increase in efficiency E_{inc} in simulations that use boundary shifting over those that maintain static simulation boundaries as a function of temperature. Results are shown for simulations on C2 with a domain size of 256×256. The results for the Ag/Ag(111) system were obtained from simulations with four PEs. The uncertainties are the standard error.

$$E' = \frac{1}{1 + c(\ln N_P)^{\beta}}, \tag{10}$$

with β ranging between 0.66 and 1.5. For our simulations, we find that β ranges between 1.49 for simulations of Ag/1-ML Ag/Pt(111) and 3.4 for the Ag/Ag(111) system. The differences between the exponents that we observe for the two systems may reflect differences in the balance between communication and computation in the parallel overhead for these systems: For Ag/1-ML Ag/Pt(111), most of the parallel overhead is consumed by computation, while communication dominates the parallel overhead for Ag/Ag(111).

C. Dynamic boundary shifting

The results in Figs. 5–8 include domain boundary shifting. Below, we discuss the impact of this feature on efficiency. Figure 9 shows the influence of boundary shifting on efficiency for both the Ag/Ag(111) and the Ag/1-ML Ag/Pt(111) systems. These results are representative of all the conditions that we probed. Here, we show the incremental efficiency E_{inc}, defined as the difference in efficiency E' between simulations run with and without boundary shifting at otherwise identical conditions. For both systems, E_{inc} is positive, indicating that boundary shifting can improve efficiency. For a fixed domain size, the gain in efficiency is apparently independent of the number of processors. We see that E_{inc} increases with increasing temperature. This trend can be at least partially attributed to increasing island rearrangement, which leads to frequent, localized motion with increasing temperature.

Interestingly, simulations of the Ag/1-ML Ag/Pt(111) system benefit more from boundary shifting than those for Ag/Ag(111). This can be understood in terms of the unique potential-energy surface (PES) topology associated with the interactions in this model. Figure 10 shows a map of the PES associated with this system during its growth at 35 K. We constructed this map by depositing atoms to a fixed coverage, then freezing the atom positions and calculating the interaction energy of a probe atom at all the vacant sites on our KMC lattice. Here, we see that favorable potential-energy basins develop as isolated attractive areas surrounded by more repulsive areas. These can be seen as green surrounded by yellow areas (or gray surrounded by light) or as yellow areas surrounded by orange areas (light surrounded by gray) in Fig. 10. Adatoms deposited near these basins become trapped and perform rapid, repeated motions. When these basins coincide with skin regions, there is a considerable loss of efficiency. Boundary shifting relocates these areas within the cores, where they cannot interfere with the timelines of their neighboring domains.

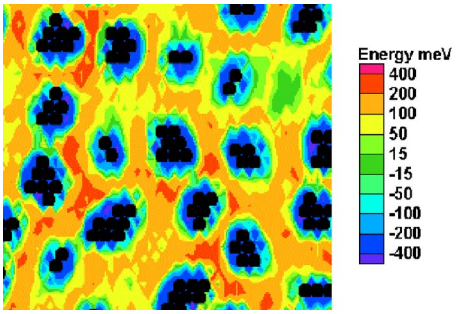

FIG. 10. (Color online) Potential-energy map of the interaction energy for simulations of Ag on 1-ML Ag/Pt(111) at 35 K. The dark dots represent adatoms and the color scale indicates the interaction energy at each vacant site on the surface.

We note that while the present boundary-shifting algorithm is beneficial for the systems simulated here, which have rapid *localized* motion, it may not be generally beneficial. For example, the present algorithm will not remedy rapid *nonlocalized* motion, as would occur for rapidly diffusing adatoms at high temperatures. The diffusion of small clusters on solid surfaces via concerted motion might also fall into this category [21]. In such systems, rapid and repeated motion in the skin is pervasive and frequent boundary shifting may actually lower the efficiency. For a moderate number of PEs, such simulations can be rendered efficient using large domain sizes, which minimize the skin-to-core area ratio, as discussed above.

IV. CONCLUSIONS

In conclusion, we presented a modified version of the SR algorithm for parallel KMC simulations of thin film growth. We showed, for two different growth models, that these simulations can achieve good efficiency–especially for large domain sizes with a moderate number of processors. By incorporating domain boundary shifting, we could gain

additional efficiency, which was substantial for the Ag on 1-ML Ag/Pt(111) system with sixth-neighbor interactions. The origins of parallel overhead, in the form of communication, computation, and idleness, depend on the computational platform employed, the system under study and, for a given system, the growth conditions, lattice size, and the number of PEs employed. Although the parallel scaling of this algorithm is poor, we note that many researchers have access to small Beowulf clusters. Implementation of the parallel SR algorithm on such platforms can be beneficial.

ACKNOWLEDGMENTS

This work was funded by NSF under Contracts No. ECC-0085604, No. DGE-9987589, and No. DMR-0514336. We acknowledge helpful conversations with Jacques Amar.

[1] S. G. Eick, A. G. Greenberg, B. D. Lubachevsky, and A. Weiss, ACM Trans. Model. Comput. Simul. **3**, 287 (1993).
[2] B. D. Lubachevsky and A. Weiss, in *Proceedings of the 15th Workshop on Parallel and Distributed Simulation* (PADS'01) (IEEE, Piscataway, NJ, 2001).
[3] Y. Shim and J. G. Amar, Phys. Rev. B **71**, 115436 (2005).
[4] K. A. Fichthorn and M. Scheffler, Phys. Rev. Lett. **84**, 5371 (2000).
[5] K. A. Fichthorn, M. L. Merrick, and M. Scheffler, Appl. Phys. A: Mater. Sci. Process. **75**, 17 (2002).
[6] K. A. Fichthorn, M. L. Merrick, and M. Scheffler, Phys. Rev. B **68**, 041404(R) (2003).
[7] M. L. Merrick and K. A. Fichthorn, Mol. Simul. **30**, 273 (2004).
[8] W. Luo and K. A. Fichthorn, Phys. Rev. B **72**, 115433 (2005).
[9] K. A. Fichthorn and W. H. Weinberg, Phys. Rev. Lett. **68**, 604 (1992).
[10] H. Brune, G. S. Bales, J. Jacobsen, C. Boragno, and K. Kern, Phys. Rev. B **60**, 5991 (1999).
[11] H. Brune, Surf. Sci. Rep. **31**, 121 (1998).
[12] H. Brune, K. Bromann, H. Roder, K. Kern, J. Jacobsen, P. Stoltze, K. Jacobsen, and J. Norskov, Phys. Rev. B **52**, R14380 (1995).
[13] K. A. Fichthorn and W. H. Weinberg, J. Chem. Phys. **95**, 1090 (1991).
[14] A. B. Bortz, M. H. Kalos, and J. L. Lebowitz, J. Comput. Phys. **17**, 10 (1975).
[15] C. K. Wong and M. C. Easton, SIAM J. Comput. **9**, 111 (1980).
[16] J. L. Blue, I. Beichl, and F. Sullivan, Phys. Rev. E **51**, R867 (1995).
[17] G. Henkelman and H. Jónsson, J. Chem. Phys. **115**, 9657 (2001).
[18] F. Montalenti, M. R. Sorensen, and A. F. Voter, Phys. Rev. Lett. **87**, 126101 (2001).
[19] O. Trushin, A. Karim, A. Kara, and T. S. Rahman, Phys. Rev. B **72**, 115401 (2005).
[20] G. Korniss, M. A. Novotny, H. Guclu, Z. Toroczkai, and P. A. Rikvold, Science **299**, 677 (2003).
[21] A. Karim, A. N. Al-Rawi, A. Kara, T. S. Rahman, O. Trushin, and T. Ala-Nissila, Phys. Rev. B **73**, 165411 (2006).