FAST KINETIC MONTE CARLO SIMULATIONS USING HASH TABLE BASED CACHING WITH APPLICATIONS TO NANOWIRE GROWTH AND SINTERING

KRISTOFER REYES†

Abstract. We describe an implementation of a rate cache designed to eliminate redundant calculations in kinetic Monte Carlo (KMC) simulations. The cache is a hash table of rates, indexed by local neighborhoods of an atom configuration. We present numerical evidence suggesting that the set of such local neighborhoods seen during KMC is small and repeating, implying that the utilization of such a cache is advantageous. We present a simulated annealing technique to search for effective, system-specific hash functions. In tandem with an optimized hash function, the implementation of the rate cache results in significant performance gains. Equipped with an efficient implementation, we are able to simulate nanowire growth by the vapor-liquid-solid (VLS) method as an example. An energy parameter study is presented. We show that the KMC model captures a wide range of observed phenomena, including faceting at the liquid-solid interface and nanowire kinking. Simulation results illustrating the role of faceting on nanowire kinking are also presented. As a second example of the implementation we study sintering, the evolution of porous granular material by surface diffusion and atomistic changes in species, representing an atom’s orientation or spin. We provide coarsening statistics and show that the mean disc-equivalent radius of the grains grows according to a power law, consistent with previous analysis and experiments.

Key words. kinetic Monte Carlo, caching, hash table, nanowire, sintering

AMS subject classifications. 82B80, 65Z99, 70-08

DOI. 10.1137/130926407

1. Introduction. Nanofabrication is the growth of nanometer scale structures and has important applications in electronic and opto-electronic devices. Because of the short length scale, such structures are grown using atomistic processes that rely on thermodynamic properties of the materials used. Such processes include surface diffusion, adsorption, and desorption. Depending on the relative rates of such processes, the resulting nanostructures can exhibit a broad range of phenomenology, leading to a corresponding variety in the macroscopically observed electronic and optical properties.

Understanding the dynamic processes inherent in the fabrication of nanostructures at an atomistic resolution is therefore important in the development of materials with desirable properties. As such, the use of atomistic modeling techniques such as kinetic Monte Carlo (KMC) complements high resolution, in situ microscopy techniques. Moreover, atomistic modeling provides users a method of varying energetic parameters such as atomic bond strengths without reference to a specific physical material, allowing the study of material properties with respect to such parameters.

Among the computational advantages of on-lattice KMC [6] is the local nature of events and their associated rates. Indeed, these properties result in efficient implementations in which sampling and updating of a configuration and rates can be done...
in sublinear time with respect to the size of the system, as measured by the number of atoms [7, 35]. Simulating on-lattice provides another type of local property that can be exploited for performance gains, and that is the subject of this paper. Specifically, the crystalline nature of the materials being simulated naturally implies repetition among neighborhoods within and between configurations visited during a KMC simulation. In this sense, we observe that there exists a neighborhood locality property in KMC. This is similar to the principle of data locality studied by computer scientists. In this context, data locality is exploited by caching: storing recently accessed blocks of memory in a low-latency cache with the expectation that such blocks will be accessed frequently.

Analogously, in this paper we describe a caching technique for storing the rates of frequently occurring local neighborhoods that arise during the normal course of simulation. Such a technique addresses an often overlooked component of KMC, which is the expensive nature of rate calculation. Albeit constant-time, this calculation must be performed several times at each Monte Carlo step, and it contributes a significant portion of floating point operations. As such, rate calculations are often a considerable bottleneck in a KMC implementation. The proposed caching procedure prevents the repeated calculation of rates, leading to a significant speed-up in the simulations. Such a speed-up allows us to simulate larger, complex systems and structures over longer time scales.

The complexity of the structures we are interested in mediates the effectiveness of the proposed rate caching technique. For simpler systems, such as a simple cubic lattice with nearest neighbor interactions and a small number of atomic species, the number of possible local neighborhoods, and hence the number of rates, is small. In this case, it is more effective to precompute the rates and store them in a static look-up table. At the other extreme are systems with broad interactions (such as models where elasticity is accounted for) or a large number of species, implying a large number of local neighborhoods. In this case, the chance of revisiting a particular neighborhood is small, diminishing the effectiveness of the rate caching technique. Therefore, the presented caching technique is applicable to intermediate systems that possess the neighborhood locality property, yet the number of possible local neighborhoods is large enough to exclude the use of direct look-up tables from consideration.

An example of how the KMC implementation presented in this paper is able to simulate larger, complex structures was presented in [34]. There, Ga droplet epitaxy and crystallization by an As flux was modeled. The simulations lend insight into the fundamental processes relevant during droplet crystallization. The paper gives an analysis of simulation results, but specific details about the implementation were omitted. The purpose of this present paper is to describe such details. In section 2, we provide a brief overview of the model used throughout the paper. An introduction to the basic KMC algorithm is presented in section 3. Section 4 reports numerical evidence of the neighborhood locality property and argues that such a property leads naturally to an effective caching scheme. A hash table is used in the caching procedure, and we describe how an efficient, application-specific hash function was obtained, namely by use of a simulated annealing search.

Additionally, section 5 presents two example applications of our KMC model and implementation as an illustration of the large-scale structures that an efficient KMC implementation can simulate. Section 5.1 describes simulations of nanowire growth by the vapor-liquid-solid (VLS) method. There, we study the effect of energy parameters that affect the rates of vapor to solid reactions and atom mobility at the liquid-solid interface, and we show how a broad range of features, including nanowire tapering and
kinking, are captured. In section 5.2, we present simulations of sintering, by which we mean the evolution of porous granular material. In this system, grains evolve by atomistic changes in orientation at grain-grain interfaces and by surface diffusion on the grain-pore interface. We show how the KMC treatment of this system is similar to that of other atomistic grain growth models, such as the Monte Carlo–Potts model, but incorporates both orientation changes and surface diffusion within a proper kinetic framework. We then present coarsening statistics and their dependence on the energy barrier associated with changes in orientation.

2. The kinetic Monte Carlo model. The underlying model used throughout this paper is an example of a bond counting, on-lattice KMC model. Most generally, atoms occupy positions on a graph \( G = (\Lambda, B) \), where \( \Lambda \) is the set of lattice positions and \( B \) are edges in the graph representing bonds between neighboring lattice positions. Throughout this paper, we consider a hexagonal lattice with periodic boundary conditions. Positions are either unoccupied or occupied with one of \( S \) atomic species, \( \sigma_1, \ldots, \sigma_S \). For example, in the nanowire simulations the different species correspond to vacuum, vapor, liquid, and solid phases, while in the sintering simulations the different species represent different orientations of the sintered material.

One of five atomistic events can occur at each step of the simulation. Atoms may desorb from the surface, adsorb onto vacant positions on the surface, diffuse along the surface (modeled by the exchange of an atom with a vacant position), exchange species with a neighboring occupied position (an atom-atom exchange), or undergo a reaction that modifies an atom’s species. All events alter a configuration by changing the species of at most two lattice positions, affecting a small number of local neighborhoods in which the two positions belong.

An energy \( E(X) \) is obtained from an atom configuration \( X \) by bond counting. Bonds between neighboring atoms \( x \) and \( y \) are assigned an energy \( \gamma(\sigma(x), \sigma(y)) \) that depends solely on the species \( \sigma(x), \sigma(y) \) of the atoms. The pairwise energies \( \gamma(\cdot, \cdot) \) are specific to the system to be simulated and are treated as parameters. The energy \( E(X) \) of an entire configuration \( X \) is the sum of all bonds within that configuration:

\[
E(X) = \frac{1}{2} \sum_{x \in X} \sum_{y \in X, \text{y a neighbor of x}} \gamma(\sigma(x), \sigma(y)).
\]

All event transitions \( X \to Y \) between states \( X, Y \) are assigned an activation energy of the form

\[
E_a(X, Y) = E(X \land Y) - E(X) + \epsilon(X, Y),
\]

where \( X \land Y \) is a well-defined intermediate state and \( \epsilon(X, Y) \) is an additional energy barrier. As will be discussed momentarily, the additional energy barrier is event and neighborhood specific. Rates for the transition \( X \to Y \) are given by

\[
r(X \to Y) = R_0 \exp \left[ -\frac{E_a(X, Y)}{k_B T} \right],
\]

where \( R_0 = 10^{13} \text{ s}^{-1} \) is a constant prefactor. Each of the five events is assigned a method for determining an intermediate state as well as an additional energy barrier.

For example, when the transition \( X \to Y \) represents an atom hopping to an adjacent, vacant position (representing the atomistic analogue of surface diffusion), the additional energy barrier \( \epsilon(X, Y) \) is set to zero, while the intermediate state is the
one obtained by removing the diffusing atom from the system. As such, the difference in bonding between $X$ and $X \wedge Y$ is precisely the bonds attached to the diffusing atom. Therefore, the activation energy $E_a(X,Y) = E(X \wedge Y) - E(X)$ represents the bonds broken in order to perform the hop. The case of two exchanging atoms is depicted in Figure 1. There, the two atoms have species $\sigma, \tau$, respectively. The intermediate state is defined by replacing the two exchanging atoms with atoms of an intermediate species, denoted $\sigma \wedge \tau$. To calculate the energy $E(X \wedge Y)$ of the intermediate state, bonding energies between the intermediate species and any other atom are obtained by averaging:

$$\gamma(\cdot, \sigma \wedge \tau) = \frac{1}{2}(\gamma(\cdot, \sigma) + \gamma(\cdot, \tau)).$$

For this exchange event, the additional energy barrier in the form (2.1) describes the extra barrier necessary to exchange two atoms and will be denoted $\epsilon_{\text{ex}}$ throughout this paper.

In the case of a reaction event in which an atom of species $\sigma$ changes to one of species $\tau$, a similar intermediate state can be defined by replacing the reacting atom with one of intermediate species $\sigma \wedge \tau$. Bonding energies between this intermediate species and any other atom is computed as in (2.3). The additional energy barrier for a reaction is denoted as $\rho$ throughout this paper.

2.1. Path connectedness. In the systems we consider, many complex morphologies and growth mechanisms are observed in experiments. For example in nanowire growth, experimentally observed phenomena such as diffusion along the length of the wire and nanowire kinking (a change in growth direction) must be captured in our model. As such, the solid-on-solid (SOS) constraint, which dictates that atoms must be positioned directly above other atoms, is insufficient. An alternative constraint must be made, however, in order to prevent the unphysical detachment of “molecular satellites” from the main system. This is illustrated in Figure 2, in which an atom hop results in a disconnected configuration. Atom configurations are hence
Fig. 3. A globally path-connected configuration on a hexagonal lattice that violates the local connectedness property. The solid line indicates a path between atoms $x$ and $y$. The local neighborhood of atom $B$ is outlined in dots.

required to be path-connected in the graph theoretic sense. That is, there must exist a chain of atoms through neighbor bonds between any pair of atoms in a configuration. Configurations containing overhangs and vacancies in the bulk are therefore possible, unlike in the SOS model, allowing our model to capture a broader range of morphologies. Any event resulting in a disconnected configuration is disallowed by assigning the rates of such events to be zero.

Path connectedness is inherently global and is hence infeasible to check efficiently. We approximate it by instead requiring connectedness on small, local neighborhoods about every atom. This is an approximation of global path connectedness, though local connectedness is not a necessary condition for the corresponding global property. This is illustrated in Figure 3. Atoms $x$ and $y$ are globally connected, but are not locally connected within the neighborhood of $x$. The size of the local neighborhood used to check this constraint affects the structures seen in the simulations. This size should be selected large enough to capture the relevant structures for the specific application being simulated. In the extreme case, the size of a local neighborhood approaches the entire domain, capturing global connectedness as a special case. Typically, the local neighborhood $B_d(x)$ about an atom $x$ is taken as the set of atoms with graph distance less than some small number, $d$,

$$
B_d(x) = \{ y \in \Lambda : d_G(x, y) \leq d \},
$$

where $d_G(x, y)$ is the graph distance on $G$.

2.2. Liquid phase. As described in the introduction, we wish to simulate systems in which there is a liquid phase. In a VLS nanowire system, nanostructural growth is mediated by the liquid phase, and so modeling reasonable physics in the liquid is important in obtaining accurate simulations. Liquid neighborhoods are identified by a simple population count. That is, within a neighborhood, the number of atoms of the species that constitute the liquid phase is tallied, and if the number of such atoms is more than some threshold volume, the neighborhood is deemed liquid. The different physics occurring within a liquid neighborhood as opposed to away from one is captured by assigning special additional energy barriers for events taking place within such a neighborhood. That is, each event $X \rightarrow Y$ can be further subcategorized according to neighborhood, and the additional energy barrier $\epsilon(X, Y)$ may be
written as
\[ \epsilon(X, Y) = \begin{cases} \epsilon^L(X, Y) & \text{if event occurs in a liquid neighborhood}, \\ \epsilon^{NL}(X, Y) & \text{otherwise}. \end{cases} \]

For example, in the nanowire simulation the vapor to solid reaction has an additional energy barrier \( \rho \). Depending on whether the reaction is to take place in the liquid or elsewhere, \( \rho \) is assigned a value \( \rho^L \) or \( \rho^{NL} \), respectively. Here we require that \( \rho^L < \rho^{NL} \). The lower energy barrier for a reaction in the liquid phase models the catalytic effect of the droplet on the vapor to solid reaction, an important feature in VLS growth.

As a matter of simplicity, the liquid is modeled on-lattice. Despite this geometric constraint, the model captures a reasonable set of physics pertinent to a liquid phase. The steps outlined above imply that liquid phases can be assigned alternative energy barriers to differentiate those events occurring within that phase. This allows us to model different diffusivities as well as the role of a liquid catalyst. Moreover, because our model is a bond counting one, the simple difference in coordination within a liquid neighborhood and outside of one induces a chemical potential between phases. Most importantly, because the model satisfies detailed balance (as described in the next section), our simulations proceed by the same thermodynamic driving force present in physical systems. That is, simulations naturally move toward configurations with lower energy. From this, thermodynamic properties of the liquid phase, such as surface tension forces at the vapor-liquid interface and solubility within liquid droplets, are emergent properties arising within the simulation.

2.3. Detailed balance. KMC is the simulation of a continuous time Markov chain (CTMC) with transition rates \( r(X \rightarrow Y) \) between configurations \( X, Y \). For those events that maintain the current canonical ensemble, that is, those events that preserve species counts, we ensure that their rates satisfy detailed balance,
\[ \pi(X)r(X \rightarrow Y) = \pi(Y)r(Y \rightarrow X), \]
where \( \pi \) is the canonical Boltzmann distribution
\[ \pi(X) = \frac{1}{Z} \exp \left[ -\frac{E(X)}{k_B T} \right] . \]

The detailed balance constraint is a sufficient condition to ensure that the equilibrium distribution of the Markov chain is \( \pi \). In this distribution, low energy states have maximum likelihood, and so the Markov chain is driven to minimize energy. This is the thermodynamic driving force behind the simulations. That the rates of the form (2.2) with activation energies as in (2.1) satisfy detailed balance is a matter of straightforward calculation, provided we impose a symmetry between intermediate states and additional energy barriers:
\[ X \land Y = Y \land X, \]
\[ \epsilon(X, Y) = \epsilon(Y, X). \]

The constraint that \( \epsilon(X, Y) = \epsilon(Y, X) \) means the additional energy barrier used in an event must be the same as the one used for its reverse event. This leads to a subtle complication when specifying alternative additional energy barriers in the liquid phase. In determining whether to apply \( \epsilon^L(X, Y) \) or \( \epsilon^{NL}(X, Y) \) to an event
X \rightarrow Y$, we must do so using a property of the affected atoms’ neighborhoods that is invariant under the event. This ensures that the same categorization will be applied for the reverse event $Y \rightarrow X$. The proposed population count outlined above is an example of such a property.

3. The KMC algorithm. As stated above, KMC is a simulation of a CTMC and may be implemented as such. Given a discrete state space of configurations, an initial state $X_0$, and rates $r(X \rightarrow Y)$ between states, the CTMC evolves according to three steps. First, a transition $X_i \rightarrow Y$ is sampled according to the probability distribution

$$P_{X_i}(Y) = \frac{1}{Z_i} r(X_i \rightarrow Y),$$

where the partition function is the summation

$$Z_i = \sum_Y r(X_i \rightarrow Y),$$

over all states $Y$ accessible from $X_i$. Second, the current configuration is altered to $X_{i+1} = Y$ according to the sampled transition. Because the transition times $X_i \rightarrow Y$ are each exponentially distributed with mean $\frac{1}{r(X_i \rightarrow Y)}$, the expected minimum waiting time is given by

$$dt_{X_i} = \frac{1}{Z_i},$$

and the simulation time is incremented by this amount. Lastly, the transition rates $r(X_{i+1} \rightarrow Y)$ and partition function $Z_{i+1}$ are retrieved or calculated in some fashion. This main loop is summarized in Algorithm 1. With the on-lattice assumption, transitions $X \rightarrow Y$ between atom configurations $X$ and $Y$ are the atomistic events outlined above and are parameterized by an atom/event pair $(x, e)$, where $e$ describes a local event of atom $x$ (e.g., $x$ desorbs, exchanges with a specific neighbor, etc.). By this local property and bond counting assumption, differences in energies, and hence rates by (2.2), are local quantities. We shall write $r(X; x, e)$ whenever we wish to emphasize the local nature of the transition rates and write $r(X \rightarrow Y)$ when the general CTMC interpretation of KMC is used.

**Algorithm 1**: The basic KMC main loop.

1. Initialize configuration $X_0$ and rateTable;
2. for $i = 0, 1, 2, \ldots$ do
3.   $(e, x) = \text{Sample(rateTable)}$;
4.   $(X_{i+1}, \text{affectedAtomList}) = \text{PerformEvent}(X_i, e, x)$;
5.   Update(rateTable, affectedAtomList, $X_{i+1}$);

If the neighborhood radius $d$ is sufficiently large, then the rates of any event involving atom $x$ can be written as functions depending on the species of atoms in $B_d(x)$, collected in an array

$$\Sigma(x) = (\sigma(y_1), \ldots, \sigma(y_n)),$$

where $y_1, \ldots, y_n$ is some linear ordering of elements in $B_d(x)$. The value of $d$ is selected so that $B_d(x)$ contains all atoms used in the calculation of any rates for
events involving $x$, in addition to the neighborhood used in determining the local
connectedness property. For the simulations in this paper, $d = 2$. The array $\Sigma(x)$ is
called the local neighborhood pattern of $x$.

Because the number of configurations is large, it is infeasible to store the entire
set of rates $r(X \rightarrow Y)$ for every pair $X, Y$ of configurations. Instead, for the current
configuration $X_i$, a subset of rates $\{r(X_i \rightarrow Y)\}$ called the rate table is
maintained. Because a transition $X_i \rightarrow X_{i+1}$ occurs in every iteration of the KMC
main loop, this rate table must be updated at each step. Since state transitions
correspond to local atomistic events, changing a configuration affects the species of
at most two lattice positions, and subsequently a small number of local neighborhood
patterns $\Sigma(x)$ for affected atoms $x$ whose neighborhoods contain the changed lattice
positions. Because the rates $r(X_i; x, e)$ are solely a function of $\Sigma(x)$, the rate tables
$\{r(X_i; x, e)\}$ and $\{r(X_{i+1}; x, e)\}$ are identical for all but a few entries corresponding
to those atoms $x$ whose neighborhood patterns have changed after the event.

There are different methods for sampling from and updating the rate table, de-
dpending on how it is maintained. A balanced approach uses a binary tree in which
leaves correspond to the rates $r(X_i; x, e)$ for every atom/event pair $(x, e)$ and each
internal node stores the sum of its two children [7]. This method is an efficient form of
inverse transform sampling [13], where sampling and updating is done by traversing
down and up branches of the tree. As such, the running time for these operations is
determined by the height of the tree, which is $O(\log N)$, where $N$ is the number of
atoms.

Previous work in addressing the efficiency of KMC has been in the proper data
structure choice of the rate table, an example of which was presented above. Another
example is the use of inverted linked-list structures to achieve constant time sampling
and updating [35]. However, profiling the execution of a KMC implementation shows
that the computation of rates contributes a significant amount to the execution time
of the implementation. This is because the calculation consists of several floating
point operations and calls to external libraries to compute rates of the form in (2.2).
Moreover, such rates must be computed several times for every step in simulations
that require up to hundreds of billions of such steps. While much work has been done
in reducing the computational complexity of sampling and updating, little attention
has been paid to the expensive, albeit constant-time calculation of rates. As KMC is
applied to more complex systems, a study of rate calculation and its impact on the
overall efficiency of a KMC implementation becomes necessary.

4. Hash table based caching. We propose that eliminating redundant rate
calculations addresses the bottleneck outlined above. As described in section 2, rates
for an atom $x$ are a function of the species pattern $\Sigma(x)$ describing the atom’s local
neighborhood. Atoms with identical species patterns are assigned identical rates. We
denote the collection of these rates as $r(\Sigma(x))$ to emphasize its dependence solely on
the species pattern, and not expressly on the atom $x$ itself. If the rates $r(\Sigma)$ of a
pattern $\Sigma$ have been previously computed during a simulation run, it is unnecessary
to recompute the rates whenever $\Sigma$ is encountered again. Instead, the rates of fre-
cquently occurring patterns should be cached for reuse. Such a caching technique would
prove effective only if the KMC algorithm revisits the same neighborhood patterns
frequently during a simulation.

As evidence showing that this repetition does indeed occur, we consider the
nanowire simulations described in section 5.1. At regular steps throughout the simula-
tion, the program determined the number of distinct neighborhood patterns $N_{\text{pattern}}$
within the current configuration. Figure 4(a) plots \( N_{\text{pattern}} \) as a function of the Monte Carlo steps \( t \), ensemble averaged over 128 identical trials. The graph shows that \( N_{\text{pattern}} \) is slowly growing and remains less than 400 neighborhoods after 400 million steps. This suggests that, within a fixed configuration, there is much redundancy between neighborhood patterns, implying spatial neighborhood locality.

While \( N_{\text{pattern}} \) measures the number of neighborhood patterns active at some time, we may also determine the cumulative number \( C_{\text{pattern}} \) of distinct neighborhoods seen up to a particular time. This is shown in Figure 4(b) as the solid line. We observe that the growth is sublinear, but approaches linear growth as time increases. Because the number of distinct neighborhoods is finite, this growth will eventually saturate, but this was not observed within the time scale considered in the simulations. The dashed line indicates the best linear fit. Its slope indicates that in equilibrium, the rate of change \( \frac{dC_{\text{pattern}}}{dt} \) is approximately \( 8.4 \times 10^{-5} \) new neighborhoods per Monte Carlo step. That is, a new neighborhood is encountered once every 12,000 Monte Carlo steps. Therefore, there is much repetition in neighborhood pattern between configurations, implying temporal neighborhood locality. The temporal and spatial locality of neighborhood patterns therefore suggests that the use of caching is indeed advantageous.

4.1. Hash table based rate caching. Because the rates of events depend on the occupancy pattern of local neighborhoods, it is desirable to have a precomputed look-up table of all the rates, indexed by pattern. Naively indexing by the species of atoms in a neighborhood results in a look-up table that is prohibitively large. In the nanowire simulations, the patterns \( \Sigma(x) \) describe the occupancy of lattice positions in \( B_2(x) \), a total of 19 atoms. Given there are 4 species and 10 events that may occur at each position, this results in a look-up table with \( 10 \times 4^{19} \approx 2^{41} \) floating point entries, though we have seen that only a small set of such entries is ever used during the simulations. One reason why the naive indexing results in a large table is that it includes neighborhoods that do not satisfy the local connectedness property, and hence will never occur in the simulations. If the number of neighborhood patterns that adhere to this constraint is sufficiently small, a clever encoding of patterns could be used as an index to the precomputed rate table. However, if there is a large number...
of species or large local neighborhoods used to check connectedness, then such an encoding method may also be infeasible. Another source of redundancy in the naive implementation is that symmetries of the lattice structure are not accounted for by a simple indexing. Such symmetries result in patterns associated with identical sets of rates, up to relabeling. Developing techniques characterizing such symmetries is system specific and depends on the number of species and the lattice used. Indeed, much work has been done in pattern recognition of local neighborhoods to address this problem [42].

We use an alternative method of characterizing commonly occurring patterns apart from the a priori large set Ω of possible patterns by storing rates of encountered patterns on-the-fly in a hash table. Previous work along these lines used a Zobrist hash to encode an atom configuration as a whole, which recovers the entire rate table associated to the configuration [28]. This method ignores the spatial and temporal neighborhood locality inherent in simulating systems of a crystalline nature in addition to the observation that, due to the local nature of a KMC transition $X \rightarrow Y$, the rate table for $Y$ is identical to that of $X$ except for a small number of entries. By encoding neighborhood patterns within a configuration rather than the configuration itself, we take advantage of the locality inherent in KMC. Stored within the hash table are portions of the rate table associated to particular neighborhood patterns, instead of the entire rate table for a given configuration. Storing by local neighborhood implies a smaller number of possible states to encode, leading to a smaller hash table, faster hash key construction, and comparison. This speed is important when placed in the context of the alternative, which is the explicit calculation of rates. The caching technique leads to a performance gain only if it can retrieve entries from the table faster than computing rates outright. This method is similar to that used in a KMC variant known as self-learning KMC, in which atomistic events (either specified explicitly or obtained from techniques such as molecular dynamics or density functional theory) are stored in a database for use in the KMC algorithm [22, 45].

When a configuration is altered, the local neighborhoods for a small number of atoms are affected, and the rates for the corresponding neighborhood patterns must be recalculated before the update algorithm is applied to the rate table. When the rates $r(\Sigma)$ of a pattern $\Sigma$ are needed, the hash table is queried to see if $r(\Sigma)$ has been previously computed. If not, it must be computed and stored in the hash table; otherwise the rates are copied to the appropriate entries in the rate table. The hash table is queried and populated by linear open addressing, using a hash function

$$h : \Omega \rightarrow \{0, \ldots, M - 1\},$$

where $M \ll |\Omega|$ is the size of the table. When the hash table is queried for rates $r(\Sigma)$, a hash index $i = h(\Sigma)$ is generated and the $i$th entry in the hash table is checked. If that entry is empty, then the pair $[\Sigma, r(\Sigma)]$ is computed and stored at entry $i$. If it is not empty, it contains a pair $[\Sigma', r(\Sigma')]$. If $\Sigma = \Sigma'$, then the rate $r(\Sigma') = r(\Sigma)$ is returned; otherwise, the hash index is incremented as $i = i + 1$, and the procedure is iterated until either $r(\Sigma)$ is found or an empty entry is encountered, in which case the rates $r(\Sigma)$ must be calculated and stored in that position. The hash algorithm is summarized in Algorithm 2. The number of times the index $i$ is incremented directly impacts the efficiency of the hashing procedure and depends on runs of nonempty entries in the table, called chains. The efficiency of Algorithm 2 depends linearly on the average chain length, so it is important to select a hash function that minimizes this quantity. In our implementation, a parameterized hash function $h = h_A$ was
Algorithm 2: Querying the rate cache by a linear, open-addressing hash table.

Input: A local neighborhood \( \Sigma \)

Output: Rates \( r(\Sigma) \)

1. Compute initial hash index, \( i = h(\Sigma) \);
2. while \( r(\Sigma) \) is not known do
   3. if \( \text{hashtable}[i] = \text{null} \) then
      4. Compute \( r(\Sigma) \);
      5. \( \text{hashtable}[i] = [\Sigma, r(\Sigma)] \);
      6. return \( r(\Sigma) \);
   7. else
      8. \( [\Sigma', r(\Sigma')] = \text{hashtable}[i] \);
      9. if \( \Sigma' == \Sigma \) then
         10. return \( r(\Sigma) \);
      11. else
         12. \( i = i + 1 \mod M \);

Fig. 5. CPU time versus the number of Monte Carlo steps performed during the nanowire simulations. Simulations without caching implemented are plotted in squares. Triangles show simulations where caching is used with an unoptimized hash function. Circles show simulation times when an optimal hash function is used.

used, where \( A \) is a set of parameters for the hash function. The parameters \( A \) were selected to minimize chain length. The selection of optimal parameters \( A \) is discussed in section 4.2. In the hash table implementation, we employ a simple hash table eviction policy, clearing the entire table when a user-specified maximum capacity is reached. As such, the number \( C_{\text{pattern}} \) is the relevant quantity in deciding the size of the table.

When implemented with an optimal hash function, the caching procedure leads to performance gains in practice. Figure 5 plots the CPU time against the number of Monte Carlo steps for several independent trails of nanowire growth simulations, varying the number of Monte Carlo steps performed and the method of caching used. Squares indicate those simulations in which no caching was used. There, when rates \( r(\Sigma) \) were required by the simulation, they were computed explicitly. The figure indicates a linear trend, and the best-fit slope suggests that without caching, simulations take about \( 14.5 \times 10^{-6} \) seconds per Monte Carlo step. Depicted by triangles are times for simulations that implement the caching procedure, but with an unoptimized choice of parameters \( A \). In this case, each simulation step requires \( 3.0 \times 10^{-6} \) seconds per
Monte Carlo step, a performance gain by a factor of about 4.8 compared to the simulations without caching. If an optimal hash function is used, the simulations take $0.9 \times 10^{-6}$ seconds per Monte Carlo step, a gain by a factor of 3 compared to the unoptimized case and a factor of 16 compared to simulations without caching. This is depicted by the circles in the figure.

In the 700 million Monte Carlo steps of a particular simulation run, there were 11.4584 billion queries into the hash table, indicating that about 16 neighborhoods were updated per Monte Carlo step on average. Of those queries, 11.4583 billion neighborhoods (99.9991%) were previously computed and stored in the hash table, requiring no extra computation. The average chain length in the hash table was approximately 1.7 nodes for the choice of optimal hash function.

4.2. Finding optimal hash functions by simulated annealing. As described above, neighborhood patterns are hashed by a hash function to produce an index into the hash table. In this context, neighborhoods $\Sigma \in \Omega$ are treated as bit strings packed into $W = |B_d| \log_2 S$ 64-bit words. Figure 6 illustrates how this is done. Each position in a neighborhood is assigned an index that linearly orders the positions of the neighborhood. The same ordering is used for every neighborhood. For example, in the figure the center atom is assigned the index 0, indicating that it is in the first position in the linear ordering. The atom coordinated to the northwest of this center atom is assigned the next index 1, indicating it is in the second position of the linear ordering. Each atom in a neighborhood is assigned a $\log_2 S$ bit identifier according to that atom’s species. These bits are then concatenated together in the linear ordering, resulting in bit strings of length $|B_d| \log_2 S$ bits. For example, the neighborhood in Figure 6 can be represented by the bit string

$$\Sigma = 10101010011001001101101010101010101010101010011,$$

assuming the 0th atom position corresponds to the leftmost two bits, etc.

The bit string is hashed by a function $h_A : \Omega \rightarrow \{0, \ldots, N - 1\}$ of the form

$$h_A(x) = \bigoplus_{i=1}^{W} (x_i \gg a_{i,1}) \oplus (x_i \gg a_{i,2}) \oplus (x_i \gg a_{i,3}) \mod M,$$

where $\oplus$ and $\gg$ are bitwise XOR and (noncyclic) right shift, respectively. The parameters $A = (a_{ij})$ are shift values such that

$$-64 < a_{ij} \leq 64,$$

with the convention that $a_{ij} < 0$ is a left shift.
Optimal shift parameters are selected with respect to a training set $\mathcal{T} \subset \Omega$ of neighborhoods. The training set $\mathcal{T}$ is generated empirically by running the simulations using an unoptimized hash function. At regular steps, the set of neighborhoods stored in the hash table is sampled uniformly, and a small proportion of this set is added to $\mathcal{T}$ as the simulation progresses. The size of $\mathcal{T}$ is chosen large enough to provide a representative sample of neighborhood patterns observed during a simulation. For example, in determining the optimal parameters for the nanowire simulations, a training set of size $|\mathcal{T}|$ over 96,000 distinct neighborhoods was used. Each choice of parameters $A$ partitions $\mathcal{T}$ into cosets 

$$
\mathcal{T}_y = \{ X \in \mathcal{T} : h(X) = y \}
$$

of configurations with identical hash value. We define a quantity $C_T$ to be the number of such cosets of size larger than 1. That is, $C_T$ is the number of configurations $X$ for which there is at least one other configuration $Y$ that hashes to the same value, i.e., a collision in the hash table. Minimizing collisions is an effective heuristic in reducing chain length, and hence we must choose parameters that minimize $C_T$.

The time to perform a brute-force search for optimal parameters over the entire space is of order $O(S^{64^d} |\mathcal{T}|)$. For the nanowire simulation, this is approximately 550 million calculations. While this brute-force calculation is feasible, we observe that the above running time scales poorly with $S$ and $B_d$. Indeed, increasing $r$ from 2 to 3 results in a brute-force calculation of approximately $2^{41} \approx 2$ trillion calculations. For sintering and grain growth simulations, in which the number of species corresponds to the number of orientations a grain may possess, $S$ can be on the order of 100,000 [14]. In this case, the brute-force calculation requires $2^{103} |\mathcal{T}|$ calculations, which is infeasible to perform.

Rather than using the brute-force calculation, good shift values are found by a simulated annealing algorithm with Metropolis selection. The simulated annealing algorithm runs as follows. Starting at a random $A = (a_{ij})$, where each $a_{ij}$ is selected uniformly in $\{-63, \ldots, 64\}$, a mutation is performed,

$$
A' = A + \Delta,
$$

where $\Delta = (\delta_{ij})$, $\delta_{ij}$ being independently and uniformly selected integers between $-4$ and 4. The mutation is kept according to a Metropolis rule based on the energy $C_T(A)$. That is, the probability $p(A \rightarrow A')$ of keeping the mutation is given by

$$
p(A \rightarrow A') = \begin{cases} 
1 & \text{if } C_T(A') < C_T(A), \\
\exp \left[ \beta (C_T(A') - C_T(A)) \right] & \text{if } C_T(A') \geq C_T(A),
\end{cases}
$$

where $\beta = \frac{1}{1000}$ represents inverse temperature (units arbitrary). The above transition probabilities satisfy detailed balance with respect to a Boltzmann distribution

$$
\pi(A) \propto \exp \left[ -\beta C_T(A) \right].
$$

Therefore, in steady state the simulated annealing algorithm will select parameters that approximately minimize $C_T(A)$. The algorithm is run for several choices of initial values.

Figure 7 shows three independent simulated annealing trials using the same training set $\mathcal{T}$ of over 96,000 distinct neighborhoods sampled during the nanowire simulations. Different, independent initial shift widths $A$ were selected for each trial. We see that the simulated annealing algorithm converges after 5000 iterations.
5. Applications. One of the first applications of KMC was in simulating the Ising model [9]. It has since been used to model a wide range of phenomena including the growth mechanisms of thin films [20, 23, 24], island nucleation and growth [2, 17], chemical vapor deposition [5], and catalytic oxidation [33]. Much work has been done incorporating elastic strain effects to simulate strained heteroepitaxial systems [3, 4, 30, 36, 37]. Off-lattice simulations include that of Al(100) crystal growth [18] and the modeling of two-dimensional (2D) cluster diffusion [22]. Apart from applications in materials science, a similar method known as dynamic Monte Carlo has been used to study protein folding [40, 41]. The model and implementation described in this paper has been previously applied to simulating Ga droplet formation and subsequent crystallization by an As flux [34].

5.1. Vapor-liquid-solid nanowire growth. We have simulated nanowire growth by the VLS method as another application of the model and implementation. In VLS growth, a liquid catalyst droplet is placed on a solid substrate. Vapor material is then introduced and is subsequently adsorbed onto the vapor-liquid interface. As the vapor is incorporated into the droplet, the liquid serves to catalyze the reaction in which vapor is converted to solid material. This solid material then nucleates at the liquid-solid interface, resulting in one-dimensional (1D) growth underneath the droplet. Common materials used in nanowire growth are silicon and liquid gold droplet as a catalyst [47]. A combination of Silane (SiCl$_4$) and hydrogen gas composes the vapor material. Within the gold material, the catalyzed reaction,

\[
\text{SiCl}_4 + 2\text{H}_2 \rightarrow \text{Si} + 4\text{HCl},
\]

results in the precipitation of Si, leading to its nucleation at the liquid-solid interface.

Experimental work in nanowires grown by the VLS method was introduced in [46]. Since then, the characterization of nanowire formation and morphology has been studied in experiments, and a wide range of phenomena has been observed. Examples of such phenomena include nanowire tapering and a change in growth direction, called kinking. For example, nanowire tapering in Si nanowires grown by the VLS method was studied in [25]. There, it was observed that suppressing the diffusion of the liquid catalyst along the nanowire side walls via the introduction of oxygen gas resulted in a negligible amount of tapering. In another study, the appearance and magnitude of tapering were shown to be controlled by vapor flux and temperature [12] in the case of self-catalyzed nanowires. Other examples of nanowire tapering include [11, 27].
Kinking in self-catalyzed GaAs nanowires can be controlled by a consideration of the deposition ratio between Ga and As [21]. Experiments have also exhibited an explicit control of nanowire kinking by vapor pressure to produce multiple kink sites on Si nanowires [44].

Some previous work regarding the simulation and modeling of nanowire growth used continuum models [38, 39]. In [39], nanowire kinking was studied in the context of faceting at the liquid-solid interface and liquid droplet statics. There, external perturbations are added to the growth of the nanowire to drive such kinking. In the KMC model, such perturbations are built-in, and we show that vulnerability to instability under such perturbations is governed by certain energetic parameters. Such unstable growth results in nanowire kinking, consistent with the theory in the continuum model. There has also been previous work in the Monte Carlo simulation of nanowires [31], which focused on the effect of experimental parameters such as temperature and deposition rate. Here we take a complementary approach, studying the effect of energetic parameters on growth kinetics and mechanisms.

In our simulations, we consider a system on a hexagonal lattice initially consisting of a liquid droplet of material L on a solid substrate of material S exposed to a vapor of material V. While not tuned to a specific physical system, the pairwise bonding energies in Table 1 and used in the simulations exhibit reasonable qualitative properties. Namely, the model specifies strong bonds (0.5 eV) within the solid material, relatively weaker bonds (0.40 eV) between liquid material, and even weaker bonds (0.35 eV) between the solid and liquid materials to encourage phase segregation. The vapor phase is considered weakly interacting with other phases, and all bonds involving a vapor atom are assigned a bonding energy of 0.10 eV. Only vapor atoms are allowed to desorb, and the additional energy barrier for desorption of vapor atoms was set to $\mu_V = 0.50$ eV.

\[
E_a(X, Y) = E(X \wedge Y) - E(X) + \rho(V, S),
\]

where $X \wedge Y$ is defined by replacing the reacting vapor atom with one of intermediate species $V \wedge S$ and $\rho(V, S)$ is the additional barrier for a reaction event. We may assign different values $\rho^L$ and $\rho^{NL}$ to $\rho$ depending on whether the reaction occurs within either a liquid neighborhood or otherwise, respectively. The energy barriers $\rho^L$ and $\rho^{NL}$ describe the additional energy barrier for reactions contributing to VLS and vapor-solid growth, respectively. The case when $\rho^{NL} > \rho^L$ therefore models the catalytic effect of the liquid droplet on the vapor-solid reaction, which is essential in VLS growth. Detailed modeling of the different reaction precursors and resultants (e.g., Cl and H in the reaction (5.1)) are not included in the model.

As an illustrative study of nanowire formation mechanisms, we examine the role of $\rho^L$ and $\epsilon_{exL}$, the additional energy barrier for atom exchanges at the liquid-solid interface. Two sets of simulations were performed. First, growth simulations were
run to study the long-term growth and macroscopic properties of nanowires over several values of $\rho^L$ and $\epsilon^{NL}_{ex}$. The initial profile consisted of a hemispherical liquid droplet of diameter 64 atoms placed on a flat solid substrate. The domain itself was 512 atoms wide, with periodic boundary conditions. Vapor material was deposited uniformly throughout the domain until 512 ML of atoms were deposited, though much of the material desorbs before the vapor-solid reaction occurs. The second set of simulations was performed to study the trajectory of individual solid atoms resulting from a vapor-solid reaction and yields microscopic insight into the mechanisms leading to the macroscopic properties observed in the first set of simulations. Here, the initial configuration is a nanowire grown as described above. Vapor atoms are then deposited as in the growth simulations, and the trajectory of the first solid atom resulting from a vapor-solid reaction is maintained until that atom becomes static, signifying its incorporation onto the wire. These trajectory simulations were run over several values of $\rho^L$ and $\epsilon^{NL}_{ex}$, and for each choice of parameters, 512 independent and identical trials were run in order to get meaningful statistics by ensemble averaging. All simulations occurred at fixed temperature $T = 350^\circ C$ and vapor deposition rate $F_V = 0.25$ ML/second.

The parameter $\rho^L$ determines the energy barrier for, and hence the rate of, a catalyzed vapor to solid reaction. Figure 8 shows typical growth simulation results for various values of $\rho^L$. In the case of low $\rho^L$ (0.75 eV), a large, broad nanowire results, in contrast with the high $\rho^L$ (1.25 eV) case, in which no growth is observed. Hence the parameter $\rho^L$ affects the growth rate and width of the nanowire. Additionally, significant tapering is observed when $\rho^L$ is small. Axial and radial growth is present in the case where $\rho^L = 0.75$ eV. When $\rho^L = 1.00$ eV, axial growth is present, but not as significant as the previous case, and radial growth is negligible. The presence and control of tapering is in qualitative agreement with what is seen in experiments, as described above.

The parameter $\epsilon^{NL}_{ex}$ affects events that occur at the liquid-solid interface, and hence describes the mobility of that interface. Among the processes that depend on $\epsilon^{NL}_{ex}$ are attachment onto, detachment from, and diffusion along the liquid-solid interface. Assuming a perfectly flat interface, the associated barriers for these events
Simulation results of nanowire growth for $\epsilon_{NL}^{ex} = 0.9, 1.10, \text{ and } 1.40 \text{ eV}$.

are

\begin{align*}
E_{\text{attachment}} &= \epsilon_{NL}^{ex} + 2\gamma(L,S) - \gamma(L,L) - \gamma(S,S) \\
&= \epsilon_{NL}^{ex} - 0.2 \text{eV}; \\
E_{\text{detachment}} &= \epsilon_{NL}^{ex} - 2\gamma(L,S) + \gamma(L,L) + \gamma(S,S) \\
&= \epsilon_{NL}^{ex} + 0.2 \text{eV}; \\
E_{\text{diffusion}} &= \epsilon_{NL}^{ex}.
\end{align*}

Changing $\epsilon_{NL}^{ex}$ therefore scales the rates for these three processes by the same amount. In the context of other processes, however, the effect of $\epsilon_{NL}^{ex}$ can be seen to alter the resulting nanowire growth modes. In particular the presence of nanowire kinking, which is observed experimentally as described above, is governed in part by this parameter. Figure 9 shows simulation results for various values of $\epsilon_{NL}^{ex}$. For a small value $\epsilon_{NL}^{ex} = 0.90 \text{ eV}$, the wire grows perpendicular to the substrate, and the liquid-solid interface is flat. There is more significant nanowire kinking in the case when $\epsilon_{NL}^{ex} = 1.10 \text{ eV}$. In this case, the liquid-solid interface is not flat and is asymmetrically faceted. When $\epsilon_{NL}^{ex} = 1.40 \text{ eV}$ there is once again little nanowire kinking, but the interface is faceted. In this case, the faceting is symmetric.

5.1.1. Diffusion-length scale, interface geometry, and kinking. The growth simulations suggest that the geometry of the liquid-solid interface affects the growth direction of a nanowire. By examining the trajectory statistics, we may gain some insight into this role. Figure 10 shows typical trajectories of solid atoms through the liquid for $\epsilon_{NL}^{ex} = 0.90, 1.10, \text{ and } 1.40 \text{ eV}$, respectively. In the case of low $\epsilon_{NL}^{ex}$, we observe that a solid atom diffuses through the liquid droplet and, upon initial attachment to the liquid-solid interface, may diffuse along that interface. When $\epsilon_{NL}^{ex}$ is increased, however, the diffusion along the interface is limited. When a solid atom attaches onto the liquid-solid interface, it is static, indicating its incorporation into the droplet. As such, the growth of the interface tends toward diffusion-limited behavior as $\epsilon_{NL}^{ex}$ is increased. For large values of $\epsilon_{NL}^{ex}$ (e.g., $\epsilon_{NL}^{ex} = 1.40 \text{ eV}$) the attachment rate is small, so a solid atom spends most of its time diffusing within the bulk of the
Fig. 10. Snapshots of trajectories for solid atoms inside the liquid droplet for $\epsilon_{\text{NL}}^{\text{ex}} = 0.9$, 1.1, and 1.4 eV, over an interval of 0.1 seconds. Each straight line segment corresponds to $9.5 \times 10^{-8}$ seconds.

Fig. 11. Snapshots of nanowire kinking and facet nucleation. (a) The final configuration. (b) Start of nanowire kinking, with flat liquid-solid interface. (c) The nucleation of a new facet. (d) Growth of the facet and droplet motion. (e) Nucleation of another facet. Here, facet directions are highlighted by solid lines.

droplet, and such trajectories are characteristically long. Indeed, the figure suggests that the solid atom visits the entirety of the liquid droplet before becoming static, avoiding the interface due to a large attachment barrier.

The simulation snapshots in Figure 9 suggest that the geometry of the liquid-solid interface plays a role in nanowire growth direction. In the case of a flat interface or growth along the VLS triple junction, the resulting nanowire grows perpendicular to the substrate, while nanowire kinking is observed in the presence of asymmetric faceting. Figure 11(a) shows another example of nanowire kinking. By examining this nanowire simulation, we can observe the evolution of the liquid-solid interface during the change in growth direction. Simulation snapshots of this process are illustrated in Figure 11, along with enlarged images of the liquid-solid interface. There, we see the nucleation of a horizontal facet (Figure 11(b)) and its growth. In Figure 11(d), the horizontal facet has become the largest along the interface, and at that stage the nanowire has changed growth directions.

We can view the kinking phenomenon as originating from the inability of the diffusion and detachment/attachment processes at the liquid-solid interface to effectively relax and flatten nucleation along that interface within a reasonable time scale. This results in the nucleation and growth of new facets. The roughness of this interface is characterized by the diffusion-length scale along the interface, which we have seen to be affected by $\epsilon_{\text{NL}}^{\text{ex}}$. When $\epsilon_{\text{NL}}^{\text{ex}}$ is small (e.g., 0.9 eV), the interface remains flat, and
no kinking is observed. When this is increased, however, the growth of new facets becomes accessible, and kinking is observed.

5.1.2. Some computational statistics. Over the entire set of growth simulations presented in this section, the average number of Monte Carlo steps required for a full simulation was 28 billion steps, while the maximum number was 117 billion steps. The corresponding average and maximum CPU times were 10 and 45 hours, respectively.

5.2. Sintering of porous granular material. As another application and illustration of the flexibility of the KMC model, we simulate sintering, a process in which a powder of material is heated below its melting point, allowing the powder to coalesce. This process may be modeled as the evolution of porous granular material, the grains signifying regions of identical crystallographic orientation. As such, the model is not unlike atomistic grain growth models such as the Monte Carlo–Potts model [48]. The fundamental difference is in the presence of vacancies within the material, resulting in the contribution of surface diffusion along the interface of such vacancies toward the evolution of the system.

Previous work in simulating sintering and growth models spans a broad range of length and time scales. Molecular dynamics simulations of sintering often involve a small number of nanoclusters of specific materials such as tungsten and TiO$_2$ [26, 29]. In [43], a kinetic Monte Carlo simulation of sintering involving only the surface diffusion of atoms between nanoclusters was studied. Larger scale simulations that include surface diffusion and grain growth were studied in [10]. There the simulations were done using Metropolis dynamics. Atomistic grain growth models that do not incorporate surface diffusion have been simulated using the Monte Carlo–Potts model and can be performed in a domain larger than the previous examples. 2D simulations in a domain consisting of 200 × 200 lattice sites have been performed [1, 19] over a time scale of up to 7000 Monte Carlo steps. More recent work simulated the model in three dimensions, within a domain of size 250 × 250 × 250 lattice sites, over a time scale of 1000 steps. In comparison, continuum models of grain growth reach even larger time scales. There, systems with up to 670,000 initial grains can be simulated over a time scale in which only 4000 grains remain [14, 15, 16].

Here, we simulate porous granular material in a 512 × 512 atom domain, with over 8000 initial grains over a period of 100 to 10,000 simulated seconds, corresponding to billions of Monte Carlo steps. In the simulations, each atom is assigned one of $S$ species, indicating a particular orientation of the atom. Pairwise bonding energies between atoms of species $\sigma_i$ and $\sigma_j$ are given by

$$\gamma(\sigma_i, \sigma_j) = \begin{cases} 
\gamma_0, & i \neq j, \\
\gamma_1, & i = j.
\end{cases}$$

Throughout this section, we fix $\gamma_0 = 0.25$ eV and $\gamma_1 = 0.50$ eV, encouraging atoms to phase segregate. Only surface diffusion and reaction events (here seen as changes in orientation) are allowed. Unlike the other two example systems we have previously considered, no special neighborhood is assigned to signify a liquid phase. As such, only one reaction energy barrier $\rho$ is specified. As before, the intermediate state for a reaction event in which an atom changes from orientation $\sigma_i$ to $\sigma_j$ is obtained by replacing that atom with one of intermediate species $\sigma_i \wedge \sigma_j$, and bonding energies involving this intermediate species are calculated according to (2.3).
As an example, we consider the event $X \rightarrow Y$ where an atom of species $\sigma$ changes orientation to species $\tau$. Suppose that atom has $n$ neighbors of orientation $\sigma$ and the remaining (assuming a hexagonal lattice) $6 - n$ neighbors are of species $\tau$. Such a reaction would occur on the boundary between two grains of orientation $\sigma$ and $\tau$, and signifies an attempted advance of the $\tau$ grain into the $\sigma$ grain. Then the activation energy of this reaction event is

$$E_a = E(X \wedge Y) - E(X) + \rho$$

$$= n\gamma(\sigma, \sigma) + (6 - n)\gamma(\sigma, \tau) - n\gamma(\sigma \wedge \tau, \sigma) - (6 - n)\gamma(\sigma \wedge \tau, \tau) + \rho$$

and the corresponding rate may be written as

$$R(X, Y) = \omega \exp \left[ -\frac{n\tilde{\gamma}}{k_B T} \right],$$

where $\tilde{\gamma} = \gamma_1 - \gamma_0$ and

$$\omega = R_0 \exp \left[ \frac{3\gamma - \rho}{k_B T} \right].$$

This rate is similar to that given in the Monte Carlo–Potts model for grain growth, with interfacial energy $\tilde{\gamma}$ and pre-exponential factor $\omega$. One key difference is in the probability of selecting such an event. Typical Monte Carlo–Potts models use the Metropolis selection criterion based on the energy difference between states $X$ and $Y$. In that model, an atom is selected uniformly to change orientation, and this change is accepted with probability 1 if it results in a decrease of energy. Otherwise, it is kept with probability proportional to that of $R(X, Y)$ above. In doing so, the Monte Carlo–Potts model ignores kinetic time scales, which is acceptable in the case where only one type of event may occur, namely orientation changes. With the KMC model, the rate of such orientation changes can be put in the context of the rates of other kinetic processes, such as surface diffusion.

### 5.2.1. Simulation results

The simulations were performed on a $512 \times 512$ hexagonal lattice with periodic boundary conditions in both directions. The initial conditions were obtained by a random sphere packing algorithm with material density specified at 80%, and is depicted in Figure 12. Each color in the figure represents one of $S = 7$ orientations an atom may possess. This number is comparable to the similar grain growth work done in [1] for the triangular lattice, finite temperature case. Simulations were performed to study the effect of $\rho$ on the coarsening time scale of the system.

The temperature $T$ was fixed at $327^\circ C$, while the parameter $\rho$ was allowed to vary between $1.25$ eV and $2.00$ eV. The system was then annealed for a total of 100 simulated seconds. In the high $\rho$ regime, surface diffusion was the dominant process, consisting of $99.9\%$ of the events when $\rho = 2.00$ eV. This drops to $70.8\%$ when $\rho = 1.25$ eV. The final configuration for each choice of $\rho$ is depicted in Figure 13. The figure suggests that the time scale for coarsening increases along with increasing $\rho$. Qualitatively, the amount of coarsening is more significant in the cases when $\rho = 1.25$ and $1.50$ eV than for the remaining two cases.

The utility of sintering is in recrystallizing material from a powder in order to grow material with a small amount of grains, minimizing the effects of grain boundary
Fig. 12. The initial configuration of the sintering simulations presented in this paper, obtained by a random sphere packing algorithm. The material density is 80%. Different colors correspond to different orientations. (See online version for color.)

Fig. 13. Final configurations of sintering simulations after 100 seconds at $T = 327^\circ C$ for $\rho = 1.25, 1.50, 1.75$, and $2.00$ eV. Different colors correspond to different orientations. (See online version for color.)

defects. As such, much experimental and analytical work on sintering and grain growth has focused on coarsening statistics, and in particular mean grain radius $\langle R \rangle$. 

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
Fig. 14. Grain coarsening statistics versus $\rho$. (Left) The number of grains versus time. (Right) The mean disc-equivalent radius $\langle R \rangle$ versus time. Best-fit power law models $\langle R \rangle = at^b + \langle R_0 \rangle$ are shown as dashed lines, and the fitted value $b$ is indicated for each plot. Each plot corresponds to a different value of $\rho$.

Such work has demonstrated the fact that the growth of this radius with respect to time scales according to a power law $\langle R \rangle \sim t^b$. For grain growth models that do not model the presence of pores in the material, and hence do not incorporate surface diffusion, analysis predicts an exponent value $b = \frac{1}{2}$ [1]. However, several experimental results yield a power law in which $b$ is between $\frac{1}{4}$ and $\frac{1}{2}$. For example, in [8] the recrystallization of lead yielded an exponent of $b = 0.4$. In [32], the sintering of cadmium oxide powder resulted in an exponent of $b = \frac{1}{3}$.

The plots in Figure 14 show two statistics of interest with respect to coarsening. The left panel presents the number of grains within a configuration as a function of simulation time, over the values of $\rho$ considered above. The plot shows a decay in this number for all values of $\rho$, though the rate of decay is more significant for $\rho \leq 1.50$. For such values of $\rho$, the number of grains quickly decays from 8600 to under 2000 within the first 10 seconds of the simulations. For these values of $\rho$, we observe from the snapshots in Figure 13 that the amount of coarsening is more substantial than the case when $\rho > 1.50$. The right panel plots the mean disc-equivalent radius $\langle R \rangle$ of the grains within a configuration. That is, $\langle R \rangle$ is the empirical mean value of the quantity $\sqrt{\frac{A}{\pi}}$, where $A$ is the area of a grain, averaged over the entire set of grains within a configuration. The radius behaves according to a power law with respect to time. Along with this empirical average radius are the best-fit power law functions of the form

$$\langle R \rangle = at^b + \langle R_0 \rangle,$$

obtained by the nonlinear least squares method. The value $\langle R_0 \rangle = 2.25$ unit lengths is the mean radius for the initial condition. For $\rho = 1.25$ and 1.50, the mean radius scales as $\langle R \rangle \sim t^{0.25 \pm 0.02}$. When $\rho = 1.75$ eV, the best-fit power law yields a value of $b = 0.52$, while $b = 0.21$ for $\rho = 2.00$.

As the number of grains decreases, the system approaches a metastable configuration. For example, consider the case where $\rho = 1.25$ eV (circles in Figure 14, left panel). Within the first 10 seconds of simulation time, the number of grains decreases from over 8000 to under 2000. After 100 seconds, the number of grains decreases to under 500. We may examine the long-term behavior in this case to understand the system’s approach to metastability. Figure 15 depicts the behavior of the system over a duration of 10,000 simulated seconds. The left panel shows the final configuration, characterized by larger grains than the system possessed after 100 seconds. The num-

number of grains within the system over the interval between 100 and 10,000 seconds is shown in the right panel. This graph shows that the number of grains decreases to under 13 grains during this time.

5.2.2. Computational statistics. The simulation depicted in Figure 15, in which 10,000 seconds were simulated, required 42 billion Monte Carlo steps and 19 hours CPU time. For the other sintering simulations, the average number of Monte Carlo steps was 4.4 billion steps while the maximum was 30 billion steps. The corresponding average and maximum CPU time was 4.6 and 15.4 hours, respectively.

6. Conclusion. As KMC is being applied to more complex systems, a careful study of efficient implementations must be made. In this paper, we presented a hash table based caching procedure designed to eliminate the calculation of transition rates for neighborhood patterns frequently seen in a simulation. In order for this procedure to be effective, a “good” hash function must be specified. Instead of explicitly constructing one, we used a parameterized form of a hash function and designed a simulated annealing algorithm to search for optimal parameters. When implemented, the hash table based caching procedure leads to a significant performance speed-up, allowing us to simulate systems such as VLS nanowire growth in a reasonable amount of time. We presented simulation results of nanowire growth that capture a wide range of observed phenomena, including tapered bases, frustrated side walls, faceting of the liquid-solid interface, and nanowire kinking. The amount and type of faceting were shown to be controlled by the parameter $\epsilon_{ex}$, and a simple model for the diffusion-length scale was developed to explain this influence. The simulations suggest that facet nucleation at the liquid-solid interface provides a mechanism for such growth modes, in agreement with past results from continuum models. Lastly, we presented simulation results of sintering, over several values of temperature and $\rho$, the additional energy barrier for changing orientations. The coarsening statistics of the grains show that the disc-equivalent mean radius behaved as a power law with respect to time. That is, $\langle R \rangle = at^b + \langle R_0 \rangle$, and depending on the choice of parameters $\rho$ and $T$, the power varied between less than 0.2 to 0.52.

Acknowledgments. The author would like to thank P. Smereka, J. Mirecki-Millunchick, T. Schulze, S. Esedoğlu, and J. Tersoff for helpful conversations.
REFERENCES


KRISTOFER REYES


