IOPscience iopscience.iop.org

Home Search Collections Journals About Contact us My IOPscience

Optimization of the Kinetic Activation-Relaxation Technique, an off-lattice and self-learning kinetic Monte-Carlo method

This article has been downloaded from IOPscience. Please scroll down to see the full text article.

2012 J. Phys.: Conf. Ser. 341 012007

(http://iopscience.iop.org/1742-6596/341/1/012007)

View the table of contents for this issue, or go to the journal homepage for more

Download details:
IP Address: 132.204.68.141
The article was downloaded on 14/02/2012 at 16:46

Please note that terms and conditions apply.

Journal of Physics: Conference Series **341** (2012) 012007 doi:10.1088/1742-6596/341/1/012007

Optimization of the Kinetic Activation-Relaxation Technique, an off-lattice and self-learning kinetic Monte-Carlo method

Jean-François Joly[1], Laurent Karim Béland[1], Peter Brommer[1], Fedwa El-Mellouhi[2] and Normand Mousseau[1]

[1] Département de Physique and Regroupement Québécois sur les Matériaux de Pointe (RQMP), Université de Montréal, C.P. 6128, Succursale Centre-Ville, Montréal, Québec, Canada H3C 3J7

[2] Science Program, Texas A&M at Qatar, Texas A&M Engineering Building, Education City, Doha, Qatar

E-mail: jean-francois.joly.1@umontreal.ca; normand.mousseau@umontreal.ca

Abstract. We present two major optimizations for the kinetic Activation-Relaxation Technique (k-ART), an off-lattice self-learning kinetic Monte Carlo (KMC) algorithm with on-the-fly event search THAT has been successfully applied to study a number of semiconducting and metallic systems. K-ART is parallelized in a non-trivial way: A master process uses several worker processes to perform independent event searches for possible events, while all bookkeeping and the actual simulation is performed by the master process. Depending on the complexity of the system studied, the parallelization scales well for tens to more than one hundred processes. For dealing with large systems, we present a near order 1 implementation. Techniques such as Verlet lists, cell decomposition and partial force calculations are implemented, and the CPU time per time step scales sublinearly with the number of particles, providing an efficient use of computational resources.

1. Introduction

Diffusion in the solid phase is dominated by rare events, with high activation energy barriers compared to system temperature. The simulation of such activated events is demanding: simulation schemes linear in time, such as molecular dynamics, often cannot attain the time scales on which those events take place. Low observed rates, however, make it possible to view the events as independent steps in a Markov chain, allowing methods such as the kinetic Monte Carlo (KMC) algorithm [1] to be used.

In the standard implementation of KMC, a predefined catalog of possible events is required, which usually dictates confining the atomic positions to a lattice. Atomic motions and the associated energy barriers can then be determined *a priori* based on local environments. However, in many systems of interest, off-lattice positions and long-range interactions play an important role in the dynamics, and standard KMC cannot adequately describe these contributions.

We recently introduced the kinetic Activation Relaxation Technique (k-ART), an on-the-fly, self-learning, off-lattice KMC method [2, 6]. K-ART combines a topological classification of local

Published under licence by IOP Publishing Ltd

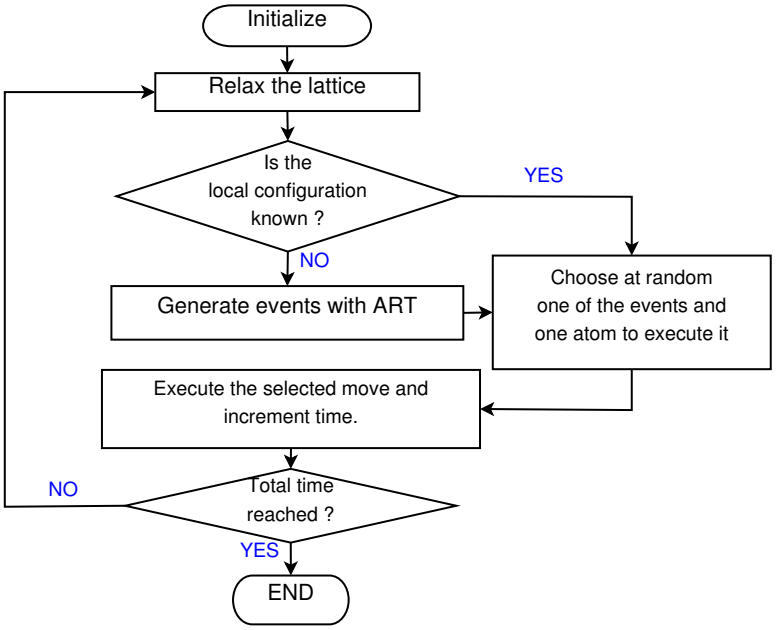

Figure 1. Flowchart showing the main structure of k-ART.

neighborhoods and events using NAUTY [3] with the *ART nouveau* method [4, 5] to find diffusion pathways as new environments are visited. Not constrained to simple geometries, k-ART has been successfully applied to a number of complex systems such as interstitials in Fe, self-defect annihilation in Si and relaxation of amorphous Si [6].

The computational complexity of the method dictates an implementation that is both effective and efficient. In this context, effectiveness means that results can be determined quickly (many results per time unit), while efficiency means that no computational resources are wasted (low computational cost per result). The nature of events and the structure of the algorithm make it possible to significantly improve the simulation through two avenues. First, the scaling of event generation with the number of particles N, which was initially order N, can be reduced to almost order one. The computational cost of k-ART becomes therefore linked only to the complexity of the structure and not its size. Second, the catalog-building part of the algorithm can be parallelized efficiently using the Message Passing Interface (MPI) [7]. These improvements are the core of this paper.

In the next sections, we describe briefly the k-ART method and its application to an ion-implanted silicon box. More details can be found in Ref. [6]. We then focus on the sublinear scaling in the number of particles and the parallelization implementation, evaluating the scaling properties both with respect to system size and number of processors used.

2. Algorithmic overview

K-ART, like conventional KMC methods, uses an event catalog to compute the rate of escape from a local minimum and to move the simulation forward in time. What is different is its self-learning and off-lattice capabilities. The algorithm consists of four steps (see also Fig. 1):

(i) At the beginning of a step, k-ART looks at each atom in the system and analyses its local topology (see Sec. 2.1). Using the program NAUTY [3], a numerical key is obtained to identify uniquely that topology. This method enables k-ART to quickly differentiate between local atomic configurations without being constrained to a fixed lattice.

(ii) For each new topology encountered, a series of event searches, using the latest version of the ART nouveau method [5, 8, 9] is launched. Generated events are analyzed and

new events are added to the database. The rate associated with the event i is given by $r_i = \tau_0 \exp(-\Delta E_i/k_B T)$, where τ_0, the attempt frequency, is fixed at the onset and, for simplification, assumed to be the same for all events (10^{13} s^{-1}). ΔE_i is the barrier height, i.e., the energy difference between saddle point and initial minimum. k_B and T are, as usual, the Boltzmann constant and the temperature respectively. These *generic events* are linked to a given topology, meaning that all atoms sharing the same local topology are considered to have access to the same set of events.

(iii) Once the event database is up to date, all active events are ordered by their activation barrier in 0.1 eV bins. Going through the bins from low to high barriers, generic events in the bin are refined on each individual atom sharing the initial topology of that event, until events representing more than 99.9% of the total rate of the system are refined. This means that low barrier events are fully relaxed in response to the actual geometry. This enables k-ART to include effects of long-ranged elastic forces. A refined event is called a *specific event* since it is associated with an individual atom. With these adjustments, the total rate catalogue is constructed.

(iv) Finally, the standard KMC algorithm [1] is applied to select an event and advance the clock. The elapsed time to the next event is computed as $\Delta t = -\ln \mu / \sum_i r_i$ where μ is a random number in the $[0, 1[$ interval and r_i is the rate associated with event i. While the rate of each event depends on a constant attempt frequency, its value was chosen to correspond to an average phonon frequency. This gives the correct order of magnitude in the calculated rates and timesteps. The clock is pushed forward, an event is selected with the proper weight and the atoms are moved accordingly, after a geometrical reconstruction. The simulation then goes back to step one and looks if any new topologies are in the system, etc.

In KMC, the residence time and the next state picked are dominated by the lowest energy barrier separating the current state from its neighbours. These low barriers can cause flickers, i.e., rapid back-and-forth movements across one or more barriers. This is generally undesirable, as it costs CPU time without leading to structural evolution, and traps the simulation in a small part of configuration space. States linked by such low barriers are called a *basin*. We developed the basin auto-constructing Mean Rate Method (bac-MRM) [6], based on the MRM by Puchala *et al.* [10], to handle these flickers and overcome the resulting dramatic slowing down of the algorithm.

The bac-MRM builds basins on-the-fly, i.e., a basin is expanded as long as the trajectory visits new states separated by low barriers. It ensures that no basin state is visited twice. While kinetics in basin are averaged out, the absorbing states are picked with the correct probability. In this way, bac-MRM permits the simulation of systems with low-energy barriers and basins, which would otherwise be inaccessible to KMC methods.

2.1. Event and topology tables

For most physical systems, the number of topologies and events can become large quite rapidly as the simulation progresses. For this reason, both the topology and the event information are kept in large and mostly sparse tables. In both cases, the table element is given by a hash key constructed from the topological information given by NAUTY.

Identifying the local topologies is the most important step in k-ART. To do so, a local cluster surrounding the atom of interest is selected (generally, all atoms situated within a sphere of given radius centered on the atom is interest) and by using a well defined distance cutoff value to determine which atoms are linked together, a connected graph is constructed. The NAUTY software takes the connected graph as input and returns an ordered set of three integers identifying the graph. The topology hash key is then calculated from the three integers characterizing a topology given by NAUTY.

For the event table, we use the hash keys of the initial, saddle and final topologies associated with an event to construct the event hash key. This way, information from both tables can be accessed quickly. In the case of hash collisions (meaning that two different topologies/events have the same final key), we just increment the key by one until a free value is found. This does not happen so often as to have a noticeable effect on the speed of the method.

This mapping supposes that there is a one to one correspondence between geometry and topology. It is normally the case, due to the constraints imposed by embedding the local graph into the global system. As discussed in Ref. [6], by construction the algorithm automatically finds out when this is not the case, allowing us to modify the graph until the correspondence holds again.

2.2. Event analysis

Since k-ART is off-lattice, a robust method for identifying diffusion pathways is needed. Every new generic (specific) event is compared to the list of already known events for that initial topology (atom). If an event with the same activation energy is already in memory, the scalar product of displacement vectors from the reference state to the final states is used to distinguish between two similar diffusion events taking place in two different directions. A second scalar product of the displacement vectors between the initial and the saddle configurations is determined, so that two events having the same end configurations, but crossing over different saddle points, can be distinguished and also treated as separate events.

3. Example: Relaxation of an ion-bombarded semiconductor

To demonstrate the versatility of k-ART, we apply the algorithm to the relaxation of a large Si box after ion implantation. In a previous work by Pothier *et al.* [11], a Si atom was implanted at 3 keV in a box containing 100 000 *c*-Si atoms at 300 K. This system was then simulated for 900 ps wit molecular dynamics. From the final configuration of this simulation, we extracted a smaller box of 26.998 atoms which contains the vast majority of induced defects and imposed periodic boundary conditions. We show the initial configuration in Fig. 2.

K-ART is applied at 300 K on this complex system. Figure 3 reports the evolution of the simulation over 282 events, which corresponds to a simulated time of 0.63 µs, at a cost of 3500 CPU-hour. During this time, the systems relaxes by 30 eV as numerous defects crystallize.

Most of the CPU time used for this run is spent on building the catalog for this new system. Even though the initial number of topologies is relatively small (839 topologies), as the system first evolves, most events generate new topologies that must also be sampled and catalogued. After 282 steps, almost 60 000 topologies (including the saddle and final minima topologies associated with potential events that are never visited but are added to the catalog) have been explored. With time, however, the catalog becomes more exhaustive and the algorithm moves faster through the phase space. Since catalogs can be merged, the information gained in preliminary simulations or runs on related systems, such as amorphous silicon, for example, can be re-used, resulting in a significant speed-up for new simulations and an optimal use of computational resources.

While standard KMC simulations easily run for millions of steps, regularly reaching time scale of minutes or more, fundamental restrictions have limited them to simple lattice-based problems such as metal on metal growth. The possibility, with k-ART, to now apply these accelerated approaches to complex materials such as ion-bombarded Si, over simulated timescale many orders of magnitude longer than can be reached by MD, opens the door to the study of numerous systems that were long considered out of reach as is demonstrated by this example.

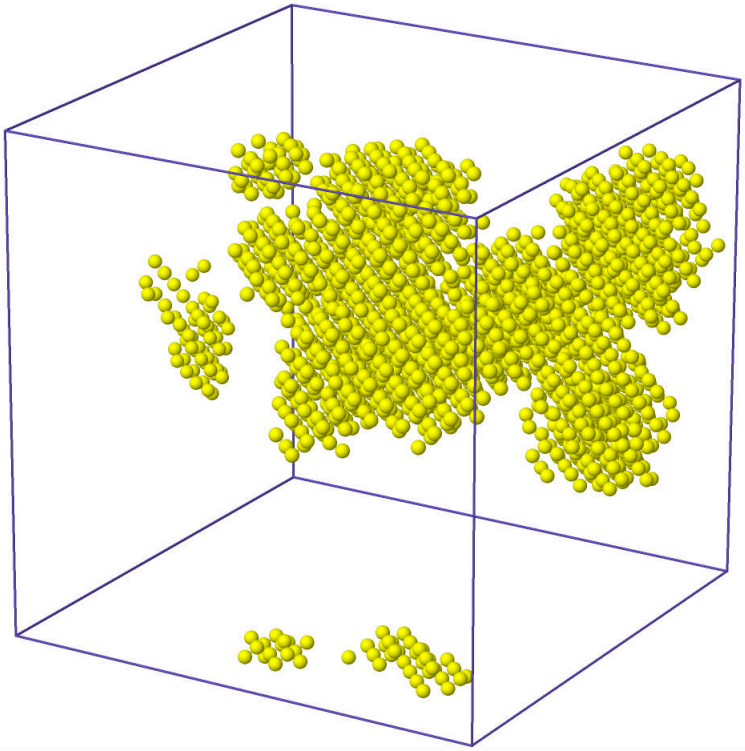

Figure 2. The initial state of our ion bombardment damage cascade simulation. We show atoms with a topology different than the perfectly crystalline topology in silicon.

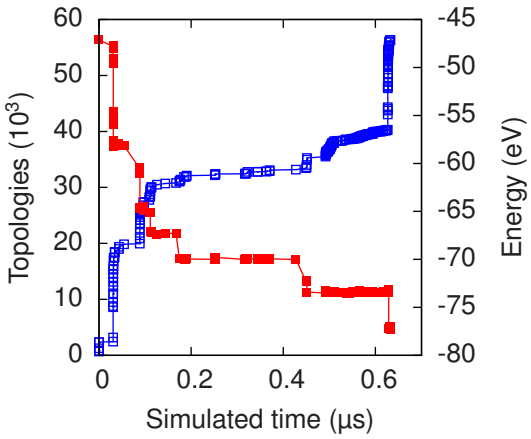

Figure 3. The number of topologies which were encountered during our simulation (□) and the energy of the system in eV (■) as a function of the simulated time

4. Optimization

4.1. Dealing with large system sizes

As shown in the previous example, systems with many tens to many hundreds of atoms to adequately capture the physics of interest. We can take advantage of the local nature of the activated dynamics to handle very efficiently these large configurations. Indeed, away from any phase transition, diffusive motion typically involves regions composed of a few tens to a few hundreds of atoms, and the forces induced by this displacement typically propagate up to a few nanometers. Because k-ART focuses on these events while ignoring the thermal vibration, we can restrict most force calculations to atoms directly involved in each event, bringing the algorithm from order N a sub-linear order..

To scale-down the computational cost associated to such large structures, we combine standard cell lists [13] and the Verlet algorithm [14] for the construction of neighbor lists with local force calculations. Starting from a table containing energy and forces for all atoms, every

force call first identifies atoms upon which forces exceed a given threshold. Forces and energy are then updated only on those *active* atoms and their first neighbours, as deformations propagate locally. With this adaptive local-force calculation, the cost of generating an event becomes almost system-size independent.

We see in Fig. 4 that the system behaves phenomenologically with an order $N^{0.4}$ scaling. While this is not quite order 1, it means that an event in a 25 000 atom box costs only 3.5 times more CPU time than one in a 1000 atom cell.

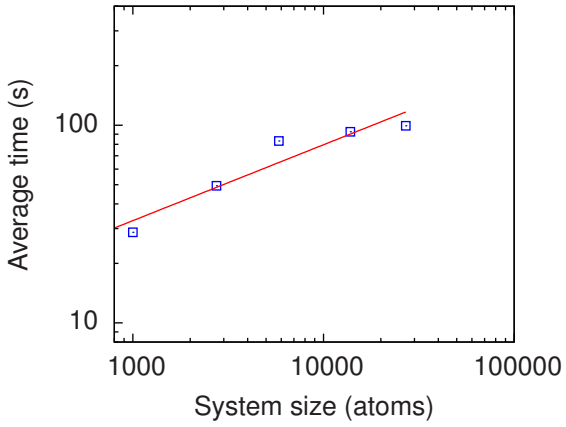

Figure 4. The mean time to attempt a generic event search for systems of various sizes. The initial configuration for each simulation was a c-Si box containing 1000 to 27000 atoms with four vacancies in each. The simulations were performed at 500 K and run on a 2.66 GHz single core on a quad-core Xeon processor. The blue squares correspond to the data, while the green line is a least-square fit of $f(N) = 2.3\,N^{0.4}$

We also ensure that computational efforts spent in a region of the system which stays unchanged from one KMC step to another are not lost. A modest amount of bookkeeping allows the program to *recycle* specific events when the local environment associated with these is kept untouched. To take into account elastic deformation, all specific saddle points are reconverged, but this step is considerably cheaper than refining from a generic conformation since the starting configuration is much closer to the real transition state.

4.2. MPI implementation

In the serial version of our algorithm, the program spends most of its time generating and refining events that are independent from each other. Our current implementation of the parallel version of k-ART focuses on optimizing this process by using a master node to perform the topological analysis and apply the KMC algorithm (steps 1 and 4) while the search for generic events and the refining of specific events (steps 2 and 3) are dispatched to workers.

For the event generation phase, the master process loops through the list of topologies to be searched and sends a search task to each available worker. The workers then apply the ART nouveau algorithm for a given atom. If the search for a saddle point and associated final state is successful, the worker checks whether the event is acceptable. Finally, the result (success or failure) is sent back to the master process. The main thread receives the information and checks whether the event is new or not. New events are then added to the event database. Every time a worker sends its results to the master, another task is sent back. This is done until the search list has been exhausted.

For the event refinement phase, the master process goes through the rate histogram as in the serial version. If events in the current energy bin need to be refined, a refinement task for each atom with the event's topology is sent to the workers. The workers refine the saddle point and send their result back (success or failure) to the master. The more events share the same energy bin (are closer in energy), the more workers processes can be used to refine them concurrently.

In our implementation, the master thread alone is responsible for all bookkeeping (tables of topologies, events, basins); the workers only need to know the starting configuration of the

search and the interaction potential, limiting communication and the memory footprint: the hash tables for a 1000-atom system require roughly one gigabyte of memory. Additionally, there is no need to keep databases in sync between master and workers. The downside is that only the master can ultimately decide if an event is accepted or rejected. This bottleneck limits the number of workers a master can control. In the following, we will explore where this limitation sets in. All calculations in this section were performed on a cluster with dual Intel Xeon E5462 quad-core processors at 3 GHz.

4.2.1. Example: bcc iron, 1019 atoms, starting from scratch In Figs. 5 and 6, the efficiency of the parallelization is displayed for the simulation of a 5-vacancy cluster in bcc iron (1019 atoms) with EAM potentials [15], starting from scratch (i.e., without event or topology catalog). In Fig. 5, the parallel efficiency $E_p = T_1/(pT_p)$ for generic event searches and specific event refinements is plotted for two different computing environments. Here, T_i is the time for one search with i processors, and p is the total number of processors. For few CPUs, the master-worker parallelization in k-ART limits the the speedup, as the master is sitting idle while waiting for results from the workers. For generic event searches, the efficiency stays competitive for a larger p, which shows that the serial part of the generic event search (evaluating newly found saddle points, update of event and topology databases) is not limiting for this system. Additionally, an average of 600 ± 40 generic event searches were performed per KMC step, so all CPUs were kept busy. For specific event refinements, the speedup T_p/T_1 saturates at around 4 for 4–8 CPUs. This is due to the fact that only 17 events needed to be refined on average at each KMC step. As the parallelization is over histogram slices, typically less than eight events can be refined in parallel, and so there is not enough work for all CPUs.

The limited efficiency of specific event refinement does not affect significantly the overall parallelization speedup. In Fig. 6, the computational cost (i.e., the total CPU time required) to perform a 40 KMC step simulation under the aforementioned conditions is displayed, normalized to the time for a serial simulation. As it is impossible to repeat identical multiprocessor simulations (the activation process is highly chaotic so even using the same seeds, the fact that we do not control on which node runs which event is sufficient to create diverging simulations), the number of generic and specific event searches fluctuates wildly: the system may remain in known areas of configuration space or it might by chance move into unexplored territory, requiring many generic event searches. To compare different runs, a typical simulation time was calculated from the average time per search and the average number of searches per run, to which the time not spent on searches was added. These numbers fluctuate much less, although there is still some randomness involved. From Fig. 6, it can be seen that, for this simulation, 8–16 CPUs will result in less than 20 % overhead compared to the serial simulation. With fewer CPUs the master process is idle for too long, while using more CPUs results in a bottleneck in the analysis on the main process. Specific events become important only for large CPU numbers (at 64 CPUs they account for about 10 % of the time spent). The program only spends a negligible amount in serial code of time outside of the event search.

4.2.2. Example: amorphous silicon, 1000 atoms, from scratch and with catalogue Amorphous silicon (a-Si) is a disordered solid that provides an interesting test model for testing parallelization. With no two atoms of our 1000-atom model sharing the same topology, a-Si represents the most challenging configuration possible for k-ART, since generic event searches need to be launched for every atom before even taking the first KMC step and convergence of the catalog is likely to require many thousands of steps.

For our test, we choose 20 searches per topology, resulting in 20 000 generic event searches for building the initial catalog. To emphasize the event search phase, we only show the results for a single KMC step. As expected, the computation cost (Fig. 8) is completely dominated

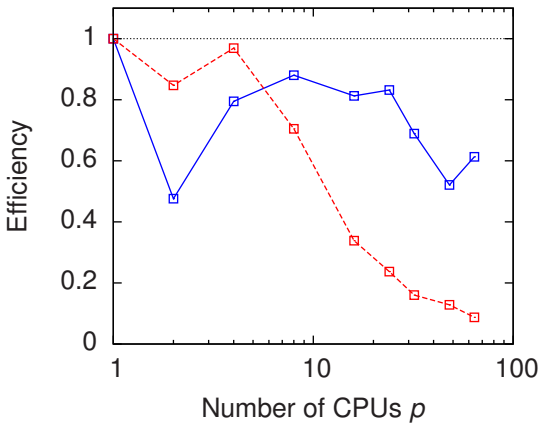

Figure 5. Parallel efficiency of event search. —— generic event search; - - - - specific event search; ····· ideal efficiency.

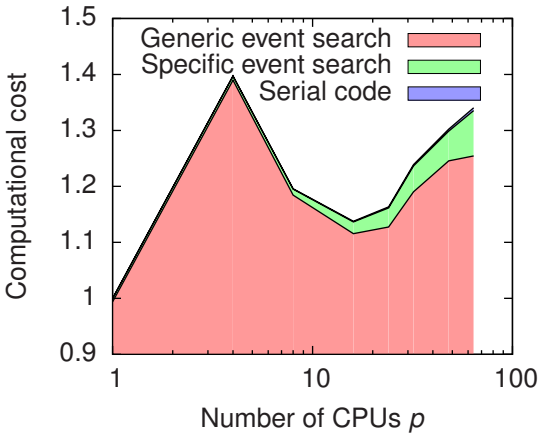

Figure 6. Computational cost (number of CPUs × per CPU time) to simulate 40 KMC events, relative to single CPU performance. Lower field: generic event search. Central field: specific event search. Top field: serial code. The relative computational cost displayed here is the inverse of the parallel efficiency.

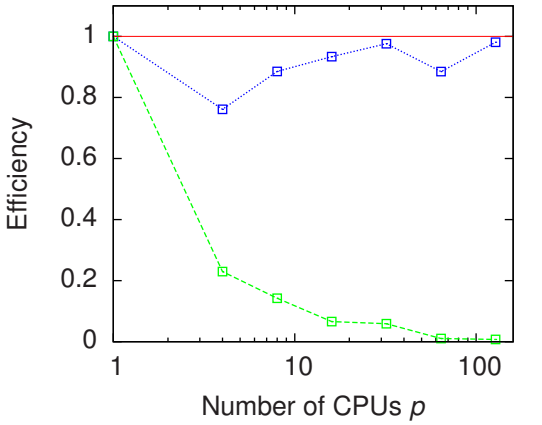

Figure 7. Parallel efficiency of event search for the first KMC step. —— generic event search; - - - - specific event search; ····· ideal efficiency.

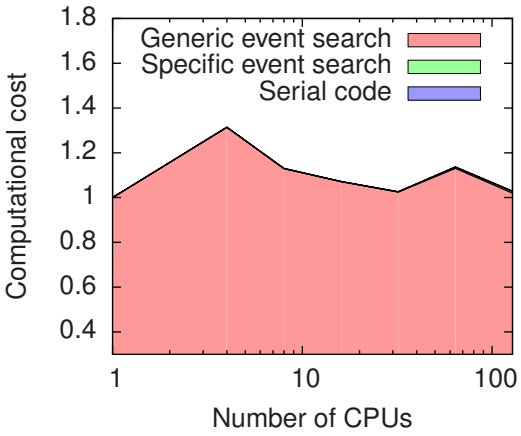

Figure 8. Computational cost to simulate the first KMC step in a-Si without catalog, relative to serial performance.

by the generic event searches and the cost of both the specific event searches and serial code is completely negligible (the small number of low energy events explain the decreasing efficiency of the specific event portion of the code after 4 CPUs). The total computational cost remains relatively constant even for a large number of CPUs. This can be seen in Fig. 7, where we see that parallelization efficiency of generic event searches is almost perfect for up to 128 CPUs. Since the efficiency is highly dependent on the ratio of the number of searches to the number of CPUs available, the result is due to the very large number of searches. This confirms that a more complex system, with more topologies to explore, will parallelize better than a simple one.

We note, moreover, that even with 128 CPUs, the event analysis bottleneck on the master

thread for this 1000-atom system is not reached; the master thread was able to cope with the increased number of searches performed per second without problem and the speed-up for this example is 125 for 128 CPUs. More CPUs could probably be added without having a large detrimental effect on the efficiency for generating the initial catalog.

5. Conclusion

We have shown that k-ART is an efficient tool to perform KMC simulations for complex system where atoms are not constrained to lattice position or where long-range elastic effects are important. By using local force calculations, our implementation scales in a near order 1 with the system size. This is necessary if the method is to be applied to larger systems with tens or hundreds of thousands of atoms. We also show that the task of performing hundreds of independent searches for events can be efficiently distributed over several CPUs.

The efficiency of the parallelization depends on the simulation phase. As the search for generic events scales better in parallelization than refining specific events, the parallel efficiency is better while there are many topologies to be explored. This will be the case either when a new simulation is started without a pre-existing catalog, or if the system evolves over time. In contrast, simulations for which an adequate catalog is available will not be able to profit similarly from parallelization. This means that parallelization is most efficient for the hardest systems, making k-ART particularly useful for characterizing the kinetic of complex problems that cannot be addressed by standard tools.

Acknowledgments

This work has been supported by the Canada Research Chairs program and by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC) and the *Fonds Québécois de la Recherche sur la Nature et les Technologies* (FQRNT). The authors thank Jean-Christophe Pothier for providing the ion bombardment simulation snapshot. We are grateful to *Calcul Québec* (CQ) for generous allocations of computer resources.

References

[1] Bortz A B, Kalos M H and Lebowitz J L 1975 *J. Comp. Phys.* **17** 10–18
[2] El-Mellouhi F, Mousseau N and Lewis L J 2008 *Phys. Rev. B* **78** 153202
[3] McKay B D 1981 *Congressus Numerantium* **30** 45–87
[4] Barkema G T and Mousseau N 1996 *Phys. Rev. Lett.* **77** 4358–4361
[5] Malek R and Mousseau N 2000 *Phys. Rev. E* **62** 7723–7728
[6] Béland L K, Brommer P, El-Mellouhi F, Joly J F and Mousseau N 2011 Kinetic activation relaxation technique submitted to Phys. Rev. E
[7] Gropp W, Lusk E and Skjellum A 1999 *Using MPI - 2nd Edition* (Cambridge, MA, USA: The MIT Press) ISBN 0–262–57132–3
[8] Marinica M C, Willaime F and Mousseau N 2011 *Phys. Rev. B* **83** 094119
[9] Machado-Charry E, Caliste D, Genovese L, Mousseau N and Pochet P 2011 *Phys. Rev. E* Accepted for publication
[10] Puchala B, Falk M L and Garikipati K 2010 *J. Chem. Phys.* **132** 134104
[11] Pothier J C, Schiettekatte F and Lewis L J 2011 *Phys. Rev. B* **83** 235206
[12] Plimpton S 1995 *Journal of Computational Physics* **117** 1–19
[13] Allen M P and Tildesley D J 1987 *Computer simulation of liquids* (New York, NY, USA: Clarendon Press) ISBN 0-19-855375-7
[14] Verlet L 1967 *Phys. Rev.* **159** 98
[15] Ackland G J, Mendelev M I, Srolovitz D J, Han S and Barashev A V 2004 *J. Phys.-Condens. Mat.* **16** S2629–S2642