Available online at www.sciencedirect.com
ScienceDirect
Mathematics and Computers in Simulation xxx (2009) xxx–xxx
MATHEMATICS AND COMPUTERS IN SIMULATION
www.elsevier.com/locate/matcom

Distributed implementation of the adaptive kinetic Monte Carlo method

Andreas Pedersen *, Hannes Jónsson

Faculty of Science and Science Institute, University of Iceland, 107 Reykjavík, Iceland

Received 17 February 2009; accepted 18 February 2009

Abstract

The program EON2 is a distributed implementation of the adaptive kinetic Monte Carlo method for long time scale simulations of atomistic systems. The method is based on the transition state theory approach within the harmonic approximation and the key step is the identification of relevant saddle points on the potential energy rim surrounding the energy minimum corresponding to a state of the system. The saddle point searches are carried out in a distributed fashion starting with random initial displacements of the atoms in regions where atoms have less than optimal coordination. The main priorities of this implementation have been to (1) make the code transparent, (2) decouple the master and slaves, and (3) have a well defined interface to the energy and force evaluation. The computationally intensive parts are implemented in C++, whereas the less compute intensive server-side software is written in Python. The platform for distributed computing is BOINC. A simulation of the annealing of a twist and tilt grain boundary in a copper crystal is described as an example application.
© 2009 IMACS. Published by Elsevier B.V. All rights reserved.

Keywords: Adaptive kinetic Monte Carlo; Long time scale dynamics; Activated transitions; Distributed computing

1. Introduction

An important challenge in computational studies of atomic scale systems is the simulation of the time evolution due to thermally activated transitions, such as configurational changes, diffusion and chemical reactions. The direct simulation of the atomic dynamics using classical equations of motion is only able to advance the system by a time increment on the order of a femtosecond for each iteration which involves an evaluation of the energy of the system and force acting on the atoms. This is due to the short time scale of atomic vibrations. As a result, the time scale accessible with reasonable computational effort using modern software and state-of-the-art hardware is on the order of nano-seconds. The time scale of thermally activated transitions in solids can, however, easily be on the order of microseconds or longer, so different algorithms are needed to reach the relevant time scale. This discrepancy in time scales between what can be simulated by discretization of the equations of motion and the time scale of thermally activated transitions is often referred to as the "time scale problem" in atomic scale simulations.

Several algorithms addressing this problem have been suggested. We focus here on the adaptive kinetic Monte Carlo [4](AKMC) method which is briefly reviewed below. Other algorithms include the parallel replica method

* Corresponding author.
E-mail address: andreas@theochem.org (A. Pedersen).

0378-4754/$36.00 © 2009 IMACS. Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.matcom.2009.02.010

Please cite this article in press as: A. Pedersen, H. Jónsson, Distributed implementation of the adaptive kinetic Monte Carlo method, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.02.010

[16], hyperdynamics [15] and temperature accelerated dynamics [12]. It is important to learn from the simulation the mechanism of activated transitions that can occur in the system without preconceived bias and it is important to not alter the rate of the transitions with the simulation method. In AKMC the computationally demanding part is to determine the possible transitions by locating saddle points on the energy surface and this can be done by a swarm of independent searches rather than just a sequential computation of time steps.

Recently, the computational capability of computers has been enhanced by increasing the computational cores in a single CPU rather than by increasing the frequency. The paradigm for supercomputing has changed similarly from being a single specifically designed and highly efficient computational unit to become clusters of computers interconnected via local high-speed network. Such clusters of computers are typically run in parallel with synchronization between instructions on different computers. For some problems, a different approach has become quite attractive, namely distributed computing where the user is not relying on a reserved and predefined computing facility but is instead able to utilize a varying number of computers connected via regular ethernet connection. A popular framework for distributed computing is BOINC that presently connects more than half a million computers that on a daily average offer 1.4 peta-flops.[1] This computational power matches the world's most powerful supercomputer and shows the enormous potential of distributed computing.

A computational task is well suited for distributed computing if it can be divided into smaller, largely independent parts which take significantly longer to execute than the extra communication introduced.

2. Adaptive kinetic Monte Carlo

The main assumption underlying AKMC is that the Transition State Theory with the harmonic approximation (HTST) applies to the system of interest. As a rule of thumb HTST is a valid and a good description of thermally activated transitions if the height of the activation energy barrier encountered is greater than ca. five times the thermal-energy, $\Delta E > 5k_BT$. In the TST approximation, the escape rate from a given initial state is estimated from the flux through a dividing surface in configuration space that separates the initial state from all possible final states. To determine such $3N-1$ dimensional TS is a difficult task. However, when the harmonic approximation is introduced the problem is simplified greatly as the dividing surface is made up of hyperplanar segments going through each of the first order saddle points on the potential energy surface with normal pointing in the direction of negative curvature. The rate of transition through each one of the hyperplanar segments is

$$k_j^{HTST} = \nu_j \ \exp\left[-\frac{E_{S,j} - E_R}{k_BT}\right] \tag{1}$$

$$\nu_j = \frac{\prod_i^D \nu_{R,i}}{\prod_i^{D-1} \nu_{S,j,i}} \tag{2}$$

where E_R and $E_{S,j}$ is the energy of the reactant state and saddle point configuration corresponding to hyperplanar segment j, ν_j is the so-called prefactor that can be expressed as the ratio between the stable vibrational modes in the reactant state, $\nu_{R,i}$ and at the saddle point $\nu_{S,j,i}$. Note that the number of vibrational modes at the saddle point is reduced by one due to the existence of one imaginary mode. Physically, the prefactor, ν_j, relates to how often the system attempts to undergo the specific mechanism.

As only the saddle point is needed to determine a specific mechanism and its rate, a table of possible events (TOE) that can take place in the system can be constructed by finding all saddle points on the potential energy rim that surrounds the energy minimum corresponding to a given state of the system. This is the approach taken in the AKMC method. To search for the saddle points, *unbiased* climbs are performed on the potential energy surface (PES). The searches are unbiased in that no preconceived notion of possible mechanism or final states is built into the search method, except the notion that the displacements of atoms will be confined to some local area (the simulated system

[1] http://boinc.berkeley.edu/.

Please cite this article in press as: A. Pedersen, H. Jónsson, Distributed implementation of the adaptive kinetic Monte Carlo method, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.02.010

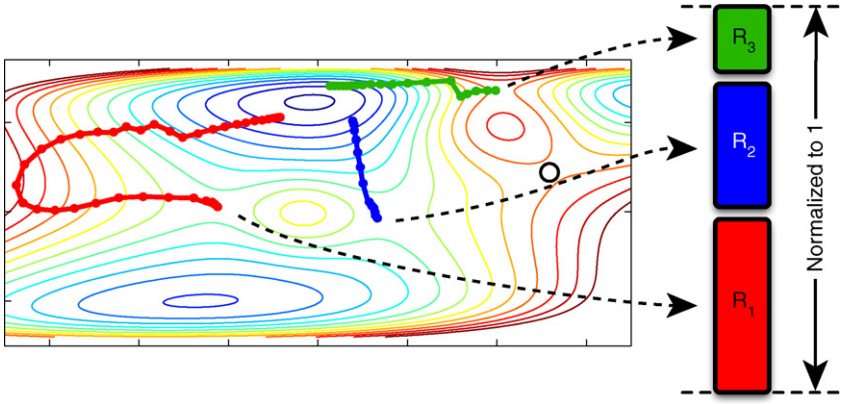

Fig. 1. Schematic of the adaptive kinetic Monte Carlo method. Left side illustrates work on the slaves where nearby saddle points on the potential energy surface are located using the minimum mode following method. Three searches are shown and the filled cirlces mark iterations where the lowest eigenmode of the Hessian is estimated using the dimer method [3]. The right side illustrates work carried out by the master. Information about the saddle points is collected, a table of the corresponding transition rates constructed and an event picked using a random number. It should be noted that the method may fail to find a saddle point, such as the one marked with the open black circle, and the corresponding transition mechanism will then be missing from the table of events.

is big enough that only a subset takes part in a transition) and the most likely regions for such transitions are those where atoms are not optimally coordinated by neighbors. After an initial random displacement of atoms in this region (typically Gaussian distributed displacements of atoms in a spherical region around an undercoordinated atom), the lowest eigenmode of the Hessian matrix is used to guide the climb up the potential energy surface and home in on a first order saddle point. This minimum-mode following method [3] is similar in spirit to earlier mode following methods [2,1] but has the advantage that second derivatives of the energy are not required and the Hessian matrix need not be constructed and diagonalized. The minimum mode can be estimated using either the Lanczos method [9] or the Dimer method [3]. A schematic representation is shown in Fig. 1. During the climb, the component of the atomic force that is parallel to the lowest eigenmode is reversed to essentially transform a first order saddle point into a minimum. The climb consists of two different phases. In the region closest to the minimum where all eigenvalue are positive, the force components perpendicular to the minimum mode (the vibrational mode corresponding to the minimum eigenvalue, λ_{min}) are zeroed whereas the parallel components are kept once the minimum mode becomes negative

$$\mathbf{F}_{eff} = \begin{cases} -(\mathbf{F}\cdot\mathbf{v}_{min})\mathbf{v}_{min} & \text{if } \lambda_{min} > 0 \\ \mathbf{F} - 2(\mathbf{F}\cdot\mathbf{v}_{min})\mathbf{v}_{min} & \text{if } \lambda_{min} < 0 \end{cases} \tag{3}$$

The system is then progressed for one step to minimize this effective force, $\mathbf{F}_{eff}$, using a standard minimizing algorithm, for example conjugate gradients.

The initial, random displacement of the atoms away from the initial, minimum energy configuration, and the lack of relaxation perpendicular to the minimum mode in the convex region, leads to sampling of a distribution of saddle points. It is important to choose the initial distribution and rate of climb in the convex region appropriately to get a good distribution of saddle points [11]. After a search has converged to a saddle point, a slight displacement along the unstable mode followed by energy minimization is used to determine the product configuration and to ensure that this saddle point corresponds to transitions from the given initial state.

When, by some criterion, the sampling is considered to be sufficiently complete the TOE can be constructed and a standard kinetic Monte Carlo algorithm used to advance the system in time by one iteration (see Fig. 1). The time increment is given by

$$\Delta t_{TST} = \frac{-\ln \mu}{\sum_j k_j^{HTST}} \tag{4}$$

where μ is a random number and j runs over all unique mechanisms. The system is then advanced to the product configuration that correspond to the chosen mechanism and this becomes the new reactant configuration. An iteration

Please cite this article in press as: A. Pedersen, H. Jónsson, Distributed implementation of the adaptive kinetic Monte Carlo method, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.02.010

of the AKMC algorithm has then been completed and a new swarm of saddle point searches is carried out from this new reactant configuration to obtain a TOE for the new state. The 'adaptive' nature of this algorithm stems from the fact that the TOE is not decided on and constructed before the simulation starts as in regular KMC, but is rather found on the fly from the energy surface during the simulation.

3. Implementation

The following section is devoted to the specific implementation of AKMC known as EON2,[2] which is a distributed code. Some parts of the code are described here and the reasoning behind certain crucial parts explained. Throughout the design and implementation phases, the following three principles were followed as far a possible:

I Transparency above speed.
II Decouple the master and the slaves.
III A fixed interface to the energy-/force-calculator.

Prioritizing transparency above speed is done to ease maintenance and reduce the occurrence of bugs. Furthermore, another highly desirable consequence is that a well written transparent code is more or less self-explaining which increases the accessibility for program developers. To decouple the master- and the slave-side is a priority in order to increase the code's robustness, as crashing or malfunctioning slaves in this case will not be able to affect the master-side and visa-versa. Another beneficial feature of separating the code in two independent parts is the opportunity to execute each side as a stand-alone program, which makes it possible to apply conventional debugging-tools. The third item enhances usability and flexibility by making it easier to introduce different potential energy functions to describe the atomic interactions. As numerous models are designed to reproduce certain characteristics of physical systems it is important not to restrain potential users to a predefined fixed set of energy-/force-calculators.

The backbone in programs for parallel- and distributed-computing is the communication scheme. Here, a master/slave approach where a single node acts as the master in charge of distributing the computations to a farm of slaves, which all focus only on the assigned tasks. This approach had already proved to be an efficient solution in an earlier implementation of AKMC.[3] The master executes the KMC algorithm and constructs the TOE on-the-fly by requesting the slaves to identify possible transition mechanisms and estimate the rates. It should be remembered that locating the saddle points is the computational demanding part. Flowcharts for the master- and slave-side are given in Figs. 2 and 3, respectively. The communication framework that is used here is BOINC.[4] It was chosen because it is now widely used for distributed computing.

EON2 is implemented using both sequential and object-oriented programming paradigms. For the master software the main parts (the KMC algorithm and communication) are written as sequential codes since it is basically just a single iterative loop that has to be executed. For the slave-side the code parts are mainly written as objects, which is done to ease substitution of the applied computational methods. As an example, the estimation of the minimum eigenmode can be done by using either Lanczos or the Dimer method. Another direct gain of the object orientation is that the interface to the force-/energy-calculators is then strictly defined automatically, which fulfills one of the main priorities for the new implementation.

The programming languages Python and C++ are used for the master- and slave-side, respectively. Table 1 lists the features, which have led to the selection of these languages. A flowchart of the master-code is shown in Fig. 2 where it should be noted that the master's task splits in two. One part is to execute the KMC algorithm using the TOE, whereas the other is to keep the slaves busy with work-units and sort the completed results that have been reported back to the server. BOINC provides a script to create new work-units for the slaves, which is schematically shown in Fig. 6. When completed work-units are reported back to the BOINC master-side they are all written into one single directory and the task of the EON2-master is therefore to inspect this directory and if results are present, determine if these are good and store them accordingly. Fig. 7 shows the directory tree where it should be noted that each reactant-state has a dedicated

[2] http://theochem.org/ andreas/installEON2.html.
[3] http://eon.cm.utexas.edu.
[4] http://boinc.berkeley.edu.

Please cite this article in press as: A. Pedersen, H. Jónsson, Distributed implementation of the adaptive kinetic Monte Carlo method, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.02.010

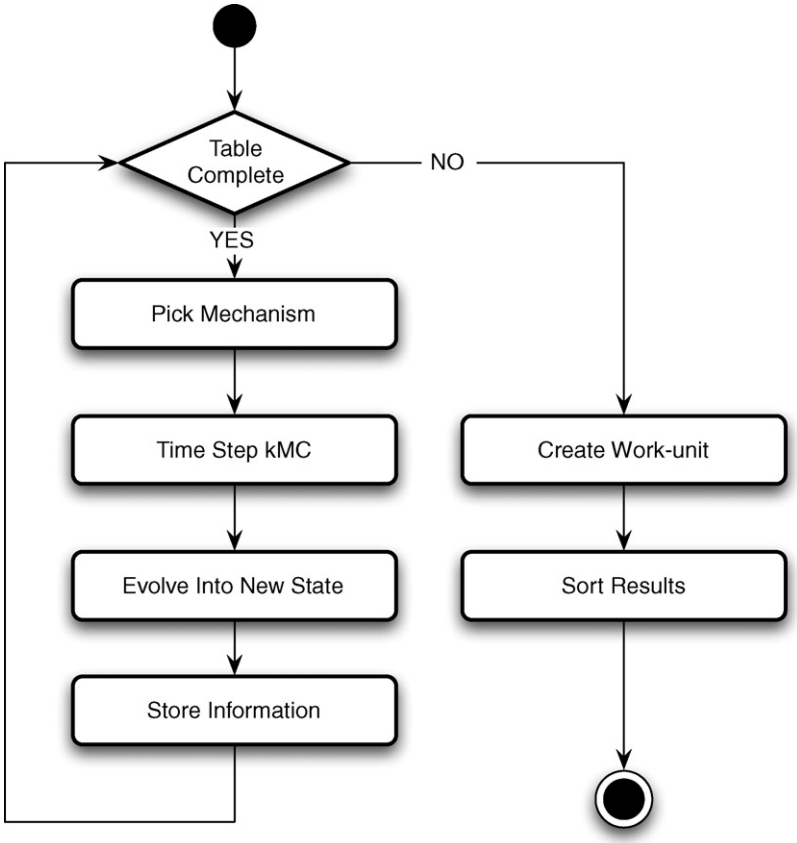

Fig. 2. Flowchart for the master-side code that has been implemented in Python. The track to the left represents the ordinary KMC algorithm. The track to the right concerns all interactions with BOINC that involve either the assignment of new work-units for the slaves (Fig. 6) or to sort results that have been reported back (Fig. 7).

sub-directory. Since a large part of the master's task is file-manipulation, Python is a good choice. It provides an easy access to the underlying operating system. For the computationally heavy slave-side, C++ is chosen since execution speed is most important and a compiled language therefore has to be used. A further reason for choosing C++ is that the slave's integration with BOINC is easy, as the BOINC slave-side code is also implemented in this language.

3.1. Slave-side

As the slave-side contains all the information about the description of the particular system being studied and computational methods, except for the KMC algorithm, this part of the code will now be explained in greater detail. To ease the exchange of data between the different classes on the slave-side the class `Matter` has been designed to serve as a data-container and its sole purpose it to carry and provide all variables needed to describe an atomic configuration. The main variables and their access methods can be found in Table 2, it should be noted that to protect the data all

Table 1
Main features of the two programming languages used in EON2. The master code is implemented in Python, whereas the computationally demanding slave code is written in C++.

	Python	C++
Run-time	Scripting	Compiled code
Pros.	Fast implementation	Fast execution
Cons.	Slow execution	Slow implementation

Please cite this article in press as: A. Pedersen, H. Jónsson, Distributed implementation of the adaptive kinetic Monte Carlo method, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.02.010

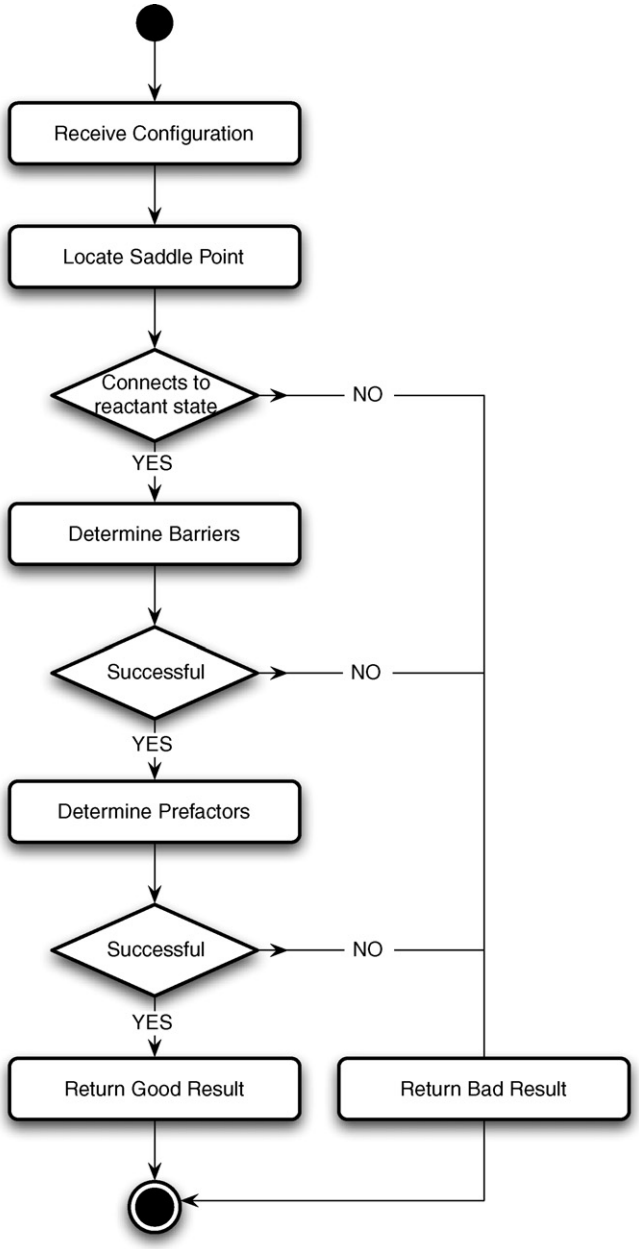

Fig. 3. Flowchart for the slave-side, which has been implemented in C++. The initial and final processes involve communication. The second process, "Locate Saddle Point", is by far is the most complex and computationally demanding part and is further illustrated in Fig. 4.

Table 2
The most important atributes of the `Matter`-class and their corresponding public access method. Besides the tabulated variables another significant attribute is a pointer to the `Potentials`-class, which makes it possible to change between the implemented force-/energy-calculators (Table 3) at run-time.

Attributes	`positions`	`isFixed`	`energy`	`forces`
Methods	`getPositions()` `setPositions()`	`getFixed()` `setFixed()`	`getEnergy()`	`getForces()`

Please cite this article in press as: A. Pedersen, H. Jónsson, Distributed implementation of the adaptive kinetic Monte Carlo method, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.02.010

Table 3
Provided force-/energy-calculators, i.e. the presently available potential functions. Furthermore, a template is given to ease interfacing to new potentials.

Ab initio	Metallic bonds	Pair potentials	Covalent bonds
VASP [a]	EMT [5]	Lennard Jones [7] Morse [10]	EDIP [6] Lenosky et al. [8] Tersoff [14] Stillinger and Weber [13]

[a] http://cms.mpi.univie.ac.at/vasp/.

variables are declared as private. Of the variables, a pointer to the `Potentials`-class is of special significance as it makes it possible at run-time to change between the implemented force-/energy-calculators listed in Table 3.

The functionality of the slave-program is to locate a nearby saddle point in an unbiased way given an initial configuration and then characterize it. The sequence of processes that are undertaken to obtain this is shown in Fig. 3. The execution is handled by the stand-alone program `EONSlave`, which calls the different processes as objects. To initialise a saddle point, search the reactant configuration and the run-time parameters are read in before the search is started.

To locate a saddle point, the responsibility is passed to the class `SaddlePoint`, which solves the task recursively as shown in Fig. 4. The search is started by displacing the reactant configuration using the `EpiCenter`-class that determines the epicenter, defined by the index of the atom around which the displacement is centered. The implemented criteria are either to pick an arbitrary free atom, to use the last atom in the configuration, one of the lowest coordinated atoms, an atom that is not FCC or HCP coordinated or an arbitrary atom from a specified list. When the reactant configuration is displaced the first recursion in the search is started by estimating the lowest value eigenmode for the Hessian in the current configuration, obtained from a request to the `LowestEigenmode`-class. In the current implementation the eigenmode is estimated using the Dimer method [3]. The system is then progressed a single step using the `Minimizers`-class accordingly to the effective force as defined in Eq. (3) before the next recursion is started.

When a search has converged to a saddle point, the two minima that correspond to initial and final states are determined by sliding down the PES along the minimum energy path. Two minimizations are performed, one is started from a configuration that corresponds to a slight displacement along the lowest eigenmode away from the saddle point, and the other corresponding to a displacement in the opposite direction. When the two minimum energy configurations have been located, the energy barrier is determined. If the barrier is smaller than a predefined maximum energy cutoff, the computationally demanding task of estimating the mechanism's prefactor is started. The estimation is done by the `Prefactors`-class where it is initially analyzed which atoms move more than a specified length due to the structural rearrangement and then finite different approximation is used to construct the Hessian for these atoms. Applying the harmonic approximation and using an Taylor expansion of the potential energy the prefactor, Eq. (1), can be expressed in terms of the mass scaled Hessians' eigenvalues, ϵ, in the reactant and saddle point configurations:

$$\nu = \frac{1}{2\pi}\sqrt{\frac{\prod_i^{} \epsilon_{R,i}}{\prod_{0<\epsilon_{SP,i}}^{} \epsilon_{SP,i}}} \tag{5}$$

With the prefactor determined, either the successful or the unsuccessful result is reported back to the master and `EONSlave` terminates. All run-time parameters are loaded and communicated via the `Parameters`-class. Among the most important parameters are the ones that determine the convergence criteria, the potential function and the distribution of the initial displacements. Besides these input parameters, the results for the barrier and prefactor are also stored in this class and this is how it is reported back to the master.

The minimization algorithm used to converge on saddle points is similar to the conjugate gradients method except that the dependence of the energy is removed and the minimizing procedure, therefore, relies only on the forces acting

Please cite this article in press as: A. Pedersen, H. Jónsson, Distributed implementation of the adaptive kinetic Monte Carlo method, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.02.010

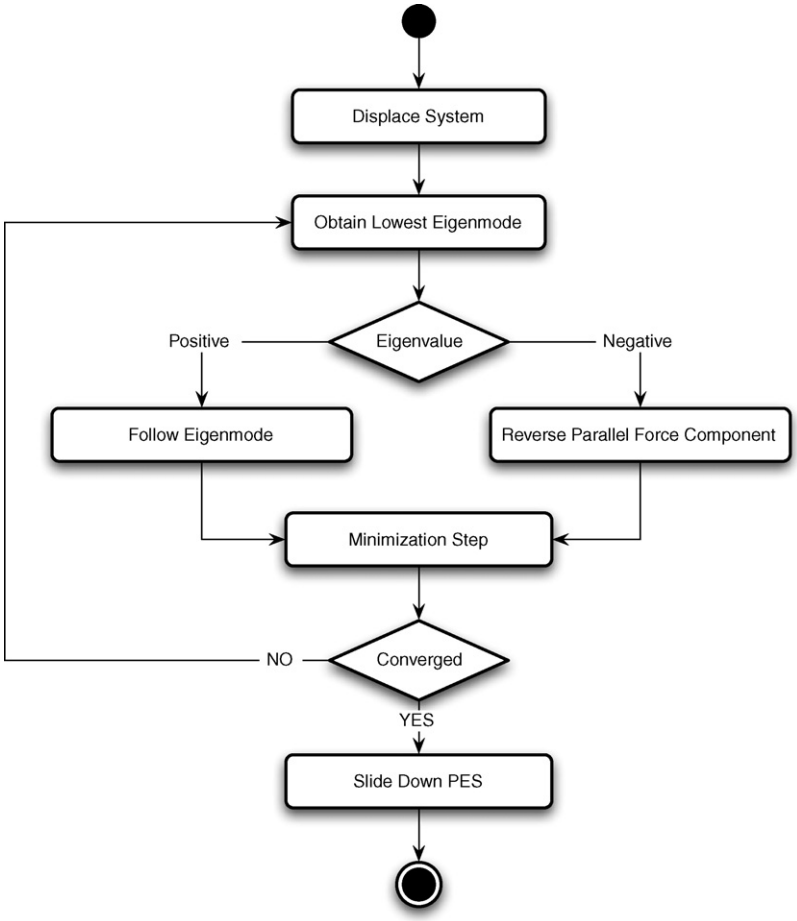

Fig. 4. Flowchart for the process "Locate Saddle Point". The recursive scheme involves the minimum mode following method. Prior the search, the configuration is displaced in order to enhance sampling of the various saddle points. The search itself is done in an unbiased manner as it is only guided by the minimum eigenmode of the Hessian at each intermediate configuration, marked by the filled cirlces in Fig. 1. Finally, it is determined which two local minima this saddle point connects by 'sliding down' the PES, i.e. minimizing after slight displacement along the mode with negative eigenvalue.

on the atoms [3].[5] Another difference is that only a single step is performed along the search-direction rather than a full line minimization. The size of this step is determined from a finite trial-step along the search-direction to estimate the change in the direction of the force

Algorithm 1. *Force Only Conjugate Gradient*
Require: The current configuration $\mathbf{r}_n$, a trial step size δr, a maximal step size Δr_{max} and the search direction $\mathbf{d}_n$.
1: Calculate $\mathbf{F}(\mathbf{r}_n)$
2: $F_{parallel} = \mathbf{F}(\mathbf{r}_n) \cdot \mathbf{d}_n$
3: Calculate $\mathbf{F}(\mathbf{r}_n + \delta r)$
4: $F_{parallel,\delta r} = \mathbf{F}(\mathbf{r}_n + \delta r \mathbf{d}_n) \cdot \mathbf{d}_n$
5: $\Delta r = \frac{F_{parallel}}{F_{parallel} - F_{parallel,\delta r}} \delta r$
6: **if** $\Delta r_{max} \leq \Delta r$
7: $\quad \Delta r = \Delta r_{max}$
end if

[5] http://eon.cm.utexas.edu.

Please cite this article in press as: A. Pedersen, H. Jónsson, Distributed implementation of the adaptive kinetic Monte Carlo method, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.02.010

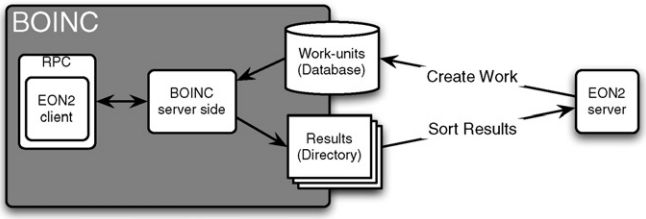

Fig. 5. Flowchart illustrating how EON2 and BOINC interact. The slave is fully encapsulated by BOINC as it creates a sandbox on a slave-machine on which the slave calculation is carried out. On the master-side, the interaction takes place via two processes: "Create Work" which adds a work-unit that is then carried out by a slave, illustrated in Fig. 6, and "Sort Result" where results received from slaves by BOINC are processed (Fig. 7).

3.2. Master/slave communication

The BOINC communication framework handles all issues regarding data transfer between the master and the slaves. It fulfils the second priority of a complete decoupling of the master and the slaves. The paradigm for the slave-execution in BOINC is Remote Process Call (RPC) which means that the slave-executable can be run as a stand-alone program. This makes debugging simpler as usual debugging-tools are applicable. In distributed execution the slave-side of BOINC creates a sandbox on the host-machine in which it executes the slave-code when a work-unit is fetched (see Fig. 5).

The BOINC master-side communication relies on an Apache web-server and a standard MySQL database (DB). This makes the communication very stable. Work is assigned to slaves by adding work-units into the DB using the script provided by BOINC as described earlier, see Fig. 6. When the work-unit is added, BOINC takes over the responsibility until the work-unit is completed and the result has been reported back. When the server-side of BOINC receives a completed result-unit it is placed in a dedicated directory accessible by the master-code, which then inspects the result-unit and moves it into its correct location (see Fig. 7).

4. Application: annealing of a grain boundary

As an illustration of the applicability of the method, a simulation of a grain boundary in copper is briefly described. The grain boundary was created by bringing together two pieces of FCC copper crystal where the two have been both twisted and rotated with respect to each other (see inset in Fig. 8). The direct simulation of classical dynamics of the system at a temperature of 300 K was first carried out for 250 ps and then for 500 ps at 135 K. An EMT potential was used to approximate the atomic interactions. The distance between the two crystal pieces normal to the grain boundary was optimized during this second annealing run. No structural change was, however, detected in the system during the second annealing. The atoms in the system are already packed tightly enough and the barrier for atomic rearrangements high enough that no thermally activated transition occurs over this time period at 135 K. The AKMC simulation is then started and after 10 transitions have been simulated, a time period of 2 ns has been covered and an annealing event found which lowers the interface energy by 1%. The total time spanned by the AKMC annealing simulation went up to 0.07 ms and if one had to perform a corresponding simulation with an ordinary classical dynamics method, this simulation would require 15 years of CPU time on a 2 GHz PowerPC. However, the AKMC simulation was completed over a period of 10 days with the computations distributed over 100 computers. This demonstrates how the AKMC

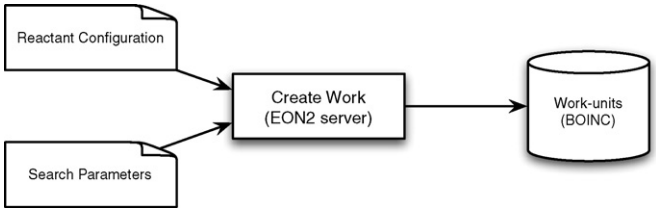

Fig. 6. Diagram showing how work-units for the slaves are created and passed into the BOINC database by the master. Each work-unit consists of an initial state configuration to which the saddle point should be connected and the run-time parameters for the climb up the potential surface. After the work-unit has been submitted to the database, BOINC takes over and hands out the task when a slave becomes available.

Please cite this article in press as: A. Pedersen, H. Jónsson, Distributed implementation of the adaptive kinetic Monte Carlo method, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.02.010

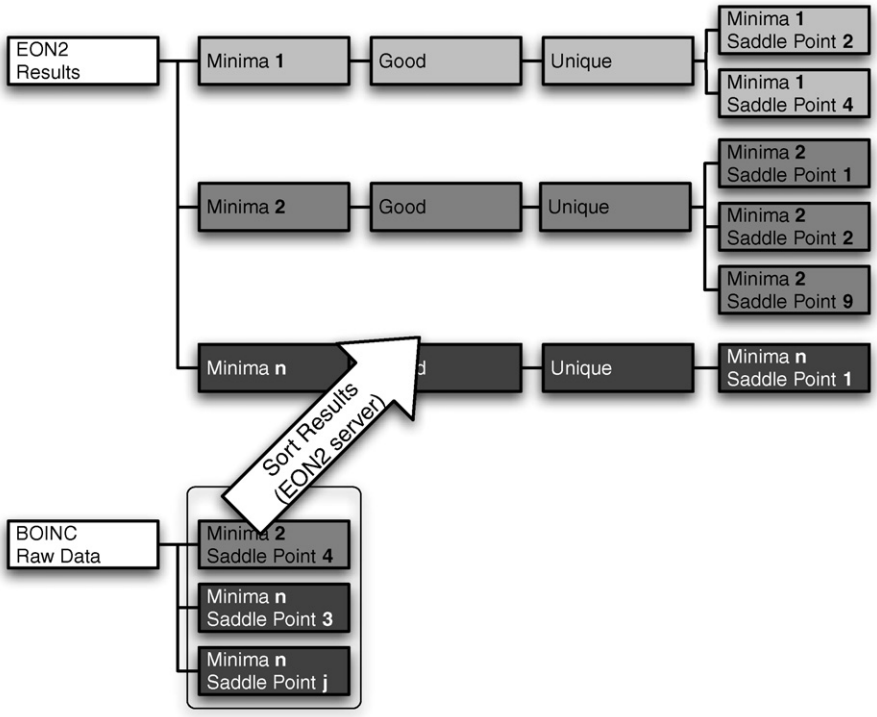

Fig. 7. Diagram for how the results received from slaves by BOINC are sorted by the master. It should be noted that since the slaves and the master are completely decoupled, results from older work-units might be received, shown as the result for "Minima 2" in the BOINC directory for "Raw Data".

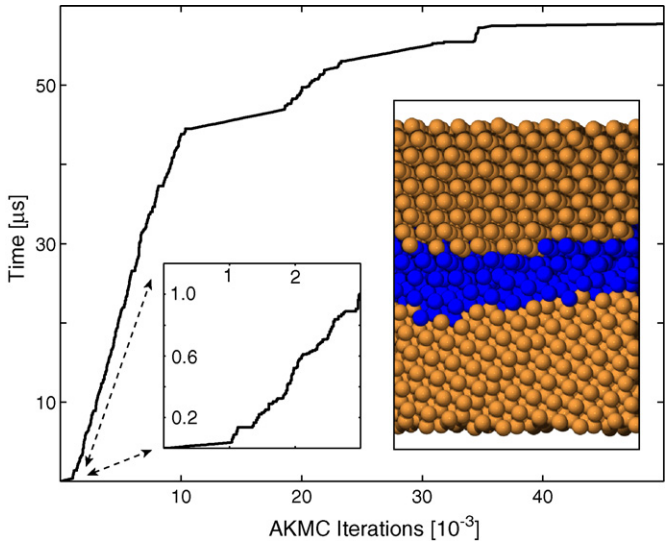

Fig. 8. Time evolution during a simulation of both a twisted and tilted grain boundary in copper. An iteration in AKMC means that a previously unknown minimum on the potential energy surface is visited and saddle point searches are farmed out to construct a table of events (TOE), or, if a previously known minimum is revisited, the TOE is reused. The time increment per iteration is calculated as in the commonly used KMC algorithm. The simulated system is shown in the inset. The time evolution is rather uneven. When saddle points with low energy are present, representing transitions with small energy barriers, the time increment for each iteration is small. After more compact configurations have been found, such low barrier events disappear and each iteration corresponds to larger time increment.

Please cite this article in press as: A. Pedersen, H. Jónsson, Distributed implementation of the adaptive kinetic Monte Carlo method, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.02.010

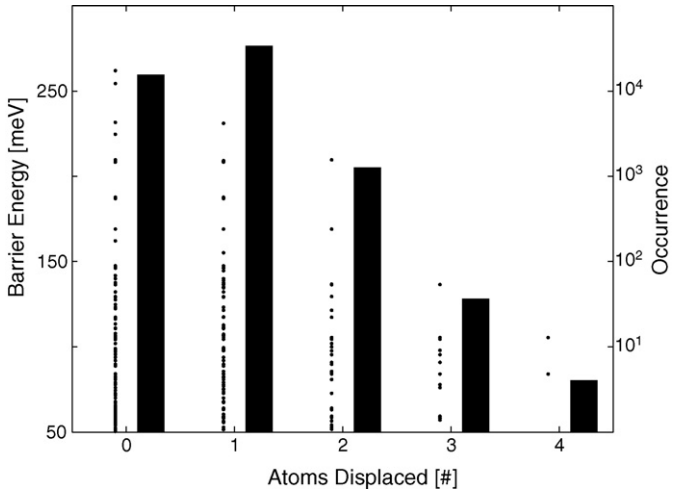

Fig. 9. Scatter plot and histogram showing how many atoms are displaced by more than 1 Å at the saddle point. The data for 0 displaced atoms accounts for the saddle points where no atom was displaced more than 1 Å. The highest energy barriers tend to involve a large displacement of just one or two atoms. Saddle points where several atoms have been displaced tend to have low energy. At some saddle points, as many as 10 atoms have been displaced by more than 0.5 Å (not illustrated here).

algorithm can help overcome the time scale problem for systems of this sort. As the simulation progressed, the energy barriers found by the saddle point searches became smaller. This ended up being a problem, as the simulation kept being stuck in a subset of states connected by low energy barriers and no significant structural change occurred during sequence of many AKMC iterations (see illustration of the time evolution in Fig. 8). We are currently implementing an extension to the method to deal with this kind of a problem.

The AKMC provides a clear illustration of the mechanism of the transitions that occur in the system during annealing. Most of these transitions involve concerted displacement of several atoms. That is, in going from an initial state minimum on the potential energy surface to a saddle point, several atoms, as many as 10, are displaced by more than 0.5 Å. The statistics for displacements that are over 1 Å are shown in Fig. 9. Two events during the simulation caused significant decreases in the energy. In the first event, the neighbors of one of the grain boundary atoms adopt a HCP-coordination around the atom. This small HCP nucleus remains stable for the remainder of the simulation and, therefore, seems to lead to stability. The second event causing a significant energy lowering involves flattening of the interface region. The event consists of 20 transitions where 10 atoms leave the grain boundary and become FCC-coordinated. As a result of the second event, the grain boundary becomes more planar. The atomic structure of the interface locally resembles a FCC(0 0 1) surface on one side and a FCC(1 1 1) surface on the other side. During the event, most activity takes place on the less stable (0 0 1)-like side. A more detailed analysis of the structure of the grain boundary and the structural transitions that occurred during the AKMC simulation is in preparation and will be published elsewhere.

5. Summary

A distributed implementation of the AKMC method for long time scale simulations of atomic systems, EON2, is described. The AKMC method makes use of the harmonic approximation of transition state theory and the key aspect of the simulation is the unbiased search for low lying saddle points on the energy surface. The saddle point searches are carried out in a distributed fashion using tens and up to hundreds of computers through the Internet. The distributed computing is based on the BOINC communication framework with full decoupling of the work carried out by master and slaves. A clean interface to the force- and energy-calculator has made it relatively easy to change from on potential function to another. The computationally demanding slave-code is implemented in C++, whereas the master is written in Python. As slaves should execute as fast as possible a compiled code is necessary. For the master, the main tasks are to keep the slaves busy by adding work-units to a data base controlled by BOINC and to sort the results returned from the slaves. Such file and data base manipulation is easily carried out with the Python language.

Please cite this article in press as: A. Pedersen, H. Jónsson, Distributed implementation of the adaptive kinetic Monte Carlo method, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.02.010

A simulation of a twisted and tilted grain boundary in a copper crystal is briefly described to illustrate the applicability of the method. There, many complex atomic transitions taking place near the grain boundary have been located by the saddle point searches, some involving significant displacement of as many as 10 atoms simultaneously.

Acknowledgements

We thank Prof. G. Henkelman for help and guidance through the earlier EON implementation, J. Schiøtz for his help in making the interface to EMT potential functions and J.-C. Berthet for valuable discussions regarding the programming. This work was supported by the European Commission DG Research (contract SES6-2006-518271/NESSHY) and by the Icelandic Research fund (adm. RANNIS).

References

[1] A. Banerjee, N. Adams, J. Simons, R. Shepard, Search for stationary-points on surfaces, Journal of Physical Chemistry 89 (1985) 52–57.
[2] C.J. Cerjan, W.H. Miller, On finding transition states, Journal of Chemical Physics 75 (1981) 2800–2806.
[3] G. Henkelman, H. Jonsson, A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives, The Journal of Chemical Physics 111 (1999) 7010–7022.
[4] G. Henkelman, H. Jonsson, Long time scale kinetic Monte Carlo simulations without lattice approximation and predefined event table, The Journal of Chemical Physics 115 (2001) 9657–9666.
[5] K.W. Jacobsen, J.K. Norskov, M.J. Puska, Interatomic interactions in the effective-medium theory, Physical Review B 35 (1987) 7423–7442.
[6] J.F. Justo, M.Z. Bazant, E. Kaxiras, V.V. Bulatov, S. Yip, Interatomic potential for silicon defects and disordered phases, Physical Review B 58 (1998) 2539–2550.
[7] J.E. Lennard-Jones, Cohesion, Proceedings of the Physical Society 43 (1931) 461–482.
[8] T.J. Lenosky, B. Sadigh, E. Alonso, V.V. Bulatov, J. Diaz, A.F. Kim, J.D. Voter, Kress, Highly optimized empirical potential model of silicon, Modelling and Simulation in Materials Science and Engineering 8 (2000) 825–841.
[9] R. Malek, N. Mousseau, Dynamics of lennard-jones clusters, Physical Review E 62 (2000) 7723–7728.
[10] P.M. Morse, Diatomic molecules according to the wave mechanics, Physical Review 34 (1929) 57–64.
[11] R.A. Olsen, G.J. Kroes, G. Henkelman, A. Arnaldsson, H. Jonsson, Comparison of methods for finding saddle points without knowledge of the final states, The Journal of Chemical Physics 121 (2004) 9776–9792.
[12] M. Sorensen, A.F. Voter, Temperature-accelerated dynamics for simulation of infrequent events, The Journal of Chemical Physics 112 (2000) 9599–9606.
[13] F.H. Stillinger, T.A. Weber, Computer simulation of local order in condensed phases of silicon, Physical Review B 31 (1985) 5262–5271.
[14] J. Tersoff, New empirical model for the structural properties of silicon, Physical Review Letters 56 (1986) 632–635.
[15] A.F. Voter, Hyperdynamics: accelerated molecular dynamics of infrequent events, Physical Review Letters 78 (1997) 3908–3911.
[16] A.F. Voter, Parallel replica method for dynamics of infrequent events, Physical Review B 57 (1998) R13985–R13988.

Please cite this article in press as: A. Pedersen, H. Jónsson, Distributed implementation of the adaptive kinetic Monte Carlo method, Math. Comput. Simul. (2009), doi:10.1016/j.matcom.2009.02.010