Parallelization comparison and optimization of a scale-bridging framework to model Cottrell atmospheres

H. Ganesan[a,b,*], C. Teijeiro[a], G. Sutmann[a,c]

[a] *Interdisciplinary Centre for Advanced Materials Simulation (ICAMS), Ruhr-University Bochum, D-44801 Bochum, Germany*
[b] *Materials Mechanics Division, Institute of Materials Research, Helmholtz-Zentrum Geesthacht, D-21502 Geesthacht, Germany*
[c] *Jülich Supercomputing Centre (JSC), Institute for Advanced Simulation (IAS), Forschungszentrum Jülich, D-52425 Jülich, Germany*

Abstract

Low carbon steels undergo strain aging when heat treated, which causes an increased yield strength that can be observed macroscopically. Such strengthening mechanism is driven by atomistic scale processes, i.e., solute segregation of carbon (C) or nitrogen interstitial atoms. Due to its low solubility, alloying elements can diffuse to defects (e.g., dislocations) and form the so-called Cottrell atmospheres. Consequently, the mobility of defects is strongly reduced because of the interaction with solutes, and higher stresses are needed to unpin them from the Cottrell atmosphere. As C segregation and atomistic motion take place at separate timescales, Classical Molecular Dynamics (MD) and Metropolis Monte Carlo (MC) are coupled in a unified framework to capture collective effects with underlying slow dynamics. The number of degrees of freedom and the need for large computational resources in this simulation require the choice of an optimal parallelization technique for the MC part of such multi-scale simulations using an unbiased sampling of the configuration space. In the present work, two different parallel approaches for the MC routine applied to the simulation of Cottrell atmospheres are implemented and compared: (i) a manager-worker speculative scheme and (ii) a distributed manager-worker over a cell-based do-

*Corresponding author
Email addresses: hariprasath.ganesan@hzg.de (H. Ganesan), carlos.teijeirobarjas@rub.de (C. Teijeiro), g.sutmann@fz-juelich.de (G. Sutmann)

main decomposition approach augmented by an efficient load balancing scheme. The parallel performance of different Fe-C containing defects with several millions of atoms is analyzed, and also the possible optimization of the efficiency of the MC solute segregation process is evaluated regarding energy minimization.
Keywords: Parallel Monte Carlo, general manager-worker, distributed manager-worker, solute segregation modeling, Fe-C system
2010 MSC: 65C05, 65Y05

---

1. Introduction

Formability is an exciting material property that determines the scope of application of engineering materials and this property depends on the ability of the material to deform plastically. In this aspect, dislocation mobility is strongly related to the plastic deformation behavior of crystalline metallic materials, and the accumulation of solute atoms in the vicinity of defects can lead to solid-solution strengthening. The light elements in the solid solution interact with the stress field introduced by a given dislocation and distribute around its core, forming the so-called Cottrell atmosphere [1], and thereby restricting the dislocation mobility during the plastic deformation. This mechanism was first observed and postulated by Cottrell and Bilby [2] as static strain aging in ferritic steels. Much research has been directed towards experimental and numerical understanding of the solute-dislocation interactions, and in some cases, it has been possible to find direct correlations to macroscopically observed material properties. In the case of bake hardened steels, which are specially used for manufacturing automobile components because of their excellent dent resistance [3], an increase in the yield strength is attributed to the interaction of carbon Cottrell atmospheres with dislocations.

In particular, the formation of carbon (C) Cottrell atmospheres in defect iron (Fe) structures is considered as a process composed of rare events, i.e., a C atom oscillates at a local minimum for a long time until it overcomes an energy barrier to find a new local minimum in the potential energy landscape. Based

on the experimentally computed diffusion coefficients, the C diffusion in ferritic steels spans in a time frame between seconds to hours [4] depending on the operating conditions (e.g., temperature or solute concentration). On the one hand, atomistic simulation approaches like Classical Molecular Dynamics (MD) are well suited for this simulation, but the integration of Newton's equations of motion at each time step [5] has a feasible timescale of only a few nanoseconds. On the other hand, Monte Carlo provides an alternative approach which is independent of underlying time scales. Therefore, Classical MD and Metropolis [6] Monte Carlo (MC) are here coupled using virtual atoms to overcome the timescale limitation. Each virtual atom is defined as a placeholder, that is, a site in the system which may potentially be suitable to place a C atom: consequently, the number of virtual atoms in a system depends strongly on the crystal structure [7]. Considering this scope, two different parallelization approaches applied for the MC simulation in the coupling scheme are compared for different scenarios according to its performance and statistical efficiency, so that the prediction capabilities of the underlying physics and the computational performance of these algorithms is evaluated. Both parallel approaches are based on the manager-worker scheme: the first approach is a general manager-worker based on a speculative scheme developed by the authors [7] and the second approach is newly proposed scheme with a distributed manager-worker that uses domain decomposition combined with a load balancing routine. The main contributions of the paper are (1) the description of the design and implementation of the distributed manager-worker, (2) it's comparative analysis with the previous general manager-worker algorithm on different ferritic systems with defects, and (3) the study of further statistical optimization possibilities for the MC procedure based on system energy data.

The rest of the work is organized as follows. Section 2 describes previous advances in parallel MC methods. Section 3 presents background information about the parallel simulation problem and describes the two parallel algorithms used in this work, with the particular focus on the distributed approach. Section 4 shows the performance comparison of the parallel algorithms for different

simulations of single crystalline metallic materials with defects. Finally, Section 5 summarizes the main outcome of the work and gives an outlook for the future work.

2. Related work

Many parallelization strategies for MC algorithms [8, 9, 10, 11, 12, 13] have generally been inspired by different short-range techniques used for MD since many years [14, 15, 16], and they have previously been proposed based on several factors: geometry, task, algorithm and target hardware architectures. Geometry parallelization is based on independent regions and suitable for applications where the inter-atomic interactions are particularly short-ranged, like for embedded-atom method (EAM) [17] potentials. A farm parallelism approach is used in work by Jones et al. [9], where a manager process performs the actual trial moves and the energy of a trial configuration is computed in parallel by worker processes. However, this approach by Jones et al. requires sequential equilibration, therefore limiting its application to large systems. In case of event parallelism, a large simulation is broken into smaller blocks of equal length executed concurrently on all workers and manager. It is particularly useful for computing the average quantities where the order of summation does not play any role. However, this approach is not suitable for the simulation of solute segregation, because each trial conformation should account for chemical and mechanical changes in the environment to mimic a sequential approach. Esselink et al. [10] devised another MC algorithm showing a chain molecule as an example, in which many trial conformations are constructed in parallel and conformation with the highest probability is chosen: nevertheless, this algorithm is most suitable for applications with very low acceptance of trial moves.

More recent implementations of parallel MC have been accomplished using a domain decomposition scheme. One simple and interesting approach by Uhlherr et al. [18] obtained very interesting results for the simulation of large polymer chains: here the system was split into active and inactive regions,

therefore ensuring a complete independence between parallel trial move executions. Considering the system under study (carbon segregation in deformed steel), executing independent sequences of individual trial moves might cause carbon trapping in the local minima (active regions), and thus limiting them from finding global minima. However, such independent sequence happens to be a smart approach for polymer systems. Sadigh et al. [13] proposed a scalable parallel MC algorithm for variance constrained semi-grand canonical ensembles with spatial domain decomposition, where the MC sample selection is performed using non-interacting moving cubes (*sampling volume*) concurrently on all processes. This approach presents interesting features regarding parallelism, but it is not effective for modeling solute segregation in the system with fixed background concentration of solutes. One main reason is that the acceptance rate tends to decline or decay for the structural minimization problems, and also solute particles might get redistributed to a specific set of processes causing a serious load balancing issue. Another parallel MC implementation for different statistical ensembles has been done by Yamakov et al. [19] for the simulation of the swap, and displacement trial moves similarly to a serial MC algorithm. Here the domain decomposition strategy is extended from the work of Sadigh et al., so that each domain has independent sampling zones and is defined using a link cell mesh for effectively identifying the neighbor particles within the prescribed interaction range. This work state the difficulty in parallelization for canonical ensemble with swap trial moves, as well as the nonavailability of an efficient algorithm to perform random swap trial moves in parallel.

In general, previous parallelization strategies are often advantageous, although limited because of the trade-off between the acceptance rate and quality of sampling, therefore being problem-specific. In this sense, the present work provides more insight on the performance of the parallel MC approach for modeling the interstitial solute segregation, and also discussing the design decisions to reduce the execution time and exploit computational resources efficiently without systematically affecting the simulation. As it is difficult to hand pick any existing parallel approach for this particular application of Cottrell atmo-

spheres modeling, because of several challenges with the different system requirements and fluctuating acceptance rate, a new scalable parallel scheme called 'distributed manager-worker' is presented, and it is compared in performance with an earlier developed scheme, referred to as 'general manager-worker' and implemented by the same authors [7].

3. Parallel implementations

The target simulations in this work are performed on ferritic (α-Fe, with bcc crystal structure) defect systems that consist of three types of atoms: Fe, C, and, virtual atoms. The Metropolis MC algorithm is implemented and executed in four main steps:

- Choose a random C atom (i.e., 'target') and a random virtual atom (i.e., 'sample') from the current system configuration.
- Perform a swap trial move between target and sample atoms by interchanging their positions.
- Compute the energy of the new configuration following the trial move.
- Evaluate the acceptance criterion by testing if the new configuration achieves lower energy than the initial configuration.

In general, the trial moves of Metropolis MC are performed sequentially because of its dependency on the previous configuration, and therefore the parallelization of the MC routine needs to adapt to this fact. In this aspect, the use of swap trial moves has been selected in order to facilitate an energy minimization by C redistribution in the simulated system, but this choice may be modified (e.g. using trial moves to insert/delete a C atom, or swapping many C atoms) without affecting the applicability or scalability of the algorithms presented in this work: the management protocols of the parallel executions is not affected by different trial moves.

First principle studies inform that a C atom prefers an octahedral site inside the bcc-Fe host matrix. Such introduced C atom creates a tetragonal lattice distortion in the ferritic host matrix and therefore exerts tension on the two first nearest neighbors and compression on the second nearest neighbors. However, the lattice distortion in the host matrix created by this impurity atom has shown to be short ranged [7], and this fact opens different possibilities for the parallelization of the MC algorithm. Each trial move involves two spherical regions in the system, which are in principle allowed to overlap, with one of the chosen atoms in the center of each region. Each individual region (sphere) can be limited in size because of the finite spatial influence of the perturbations induced by the particle swap. Therefore, the relaxation and energy computation of each sphere is local and provides a natural possibility for parallel computation. In the case of ferritic defect systems, the atomic interactions around the chosen atoms (i.e., C and virtual atom) involved in every trial move are defined within a sphere radius of 2 nm with an additional layer of influence in which only the atom energy may vary.

Figure 1 gives an example of the type of systems simulated by the coupling scheme of MD with MC: a ferritic single crystal with virtual atoms introduced at all the octahedral sites and containing defect networks. The simulation box includes a fixed concentration of solutes (in this case, C atoms) randomly distributed. As indicated previously, at each MC step a random C and virtual atom are chosen, and their positions are exchanged (i.e., the particle species are swapped). Two spherical volumes are constructed according to the given cut-off radius using the coordinates of both selected particle as centers. The energies of these spherical volumes are computed before and after the trial move using Molecular Statics simulations via MD interface using the IMD library [20]. Based on the successful outcome of the evaluated acceptance criterion (i.e., the energy of the spheres after the trial move simulation is lower than before), all atomic entities are updated within the spheres.

In the next subsections, we discuss the two parallel MC algorithms, i.e., general and distributed manager worker.

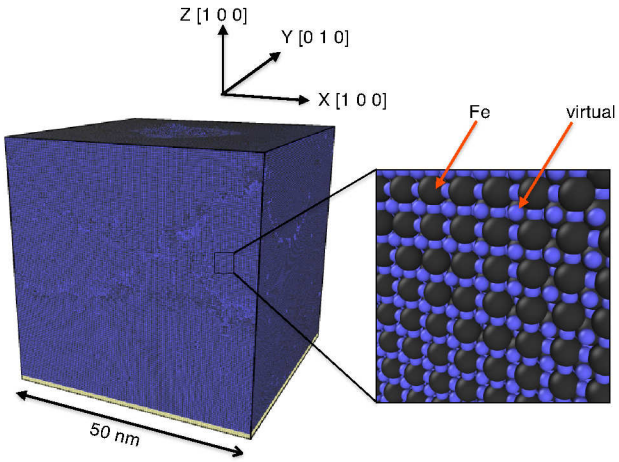

Figure 1: Nanoindented ferritic single crystal with the lateral length $L = 50$ nm. Zoomed-in configuration showing the host Fe atoms and virtual atoms sitting at the octahedral sites.

3.1. General manager-worker approach

The general manager-worker code for MC has been implemented in C++ using the MPI library [21] for communications. It consists of one manager process, which administrates the whole simulated system, and several worker processes, which execute the tasks demanded by the manager. The fast exploration of the configuration space requires efficient sampling, together with fast energy computation. For general manager-worker, the distance between two selected particles for swap type trial move is not restricted by finite distance. Meaning that the configuration space is more effectively explored in a non-local way. Unbiased sampling is realized by choosing the C and virtual atoms to be swapped using the uniform probabilities $P_c = \frac{1}{N_c}$ and $P_v = \frac{1}{N_v}$ respectively, where N_c and N_v are defined as the number of C and virtual atoms in the global system. After that, the local regions around the target atoms are relaxed within a radius $R = 2$ nm and a new possible value for the local energy minimum is computed. Following the principle of localized energy computations in spherical

subsystems, non-overlapping spheres in the system may be executed as independent MC trial moves in parallel [7], using the cut-off radius for the definition of spheres as the reference.

This parallel implementation of Metropolis MC uses a speculative algorithm, in which the overlap between spheres of different trial moves is allowed, and conflicts are resolved after their execution. According to the requested number of trial moves for a given simulation, the manager creates as many jobs as necessary for the worker processes and appends them to a job queue in the order of construction. In the case of spatial overlap between spheres that belong to different trial moves executed by different worker processes, the manager solves the conflict after the execution of the affected jobs and according to the step identifier of the job. For example, if a job i has been generated and accepted, the system configuration needs to be updated, and therefore later jobs with identifier j ($j > i$) that have a conflict with i and have started their execution before i has finished must necessarily be discarded. In the event of high acceptance rate, there could be an increased discard rate and thus affecting the parallel performance. However, the overlapping probability is low when the sampling regions are separated by distance larger than the influence range e.g., homogeneous C distribution. The job queue is dynamic and any completed, or processed job in the queue is pulled out of the queue in a FIFO order until no jobs are left in the queue, which indicate the end of the simulation. Any discarded job in the job queue needs to be repeated until it becomes a valid MC step (i.e., accepted or rejected).

A simple scheme of the general Manager-worker approach is illustrated in Figure 2 with one manager and three worker processes. The different spherical volumes represent the parallel trial moves, and the number of the spheres (see Fig., 2) corresponds to the MC step (i.e., the ordered sequence of trial moves generated by the manager). Here the created jobs are assigned to target worker processes in the order of request queue, without any relation between worker identifier and MC step. The image also illustrates a spatial conflict between spheres 1 and 3, which needs to be solved by the manager after the execution of

both trial moves and may cause the job 3 to be discarded if job 1 is accepted.

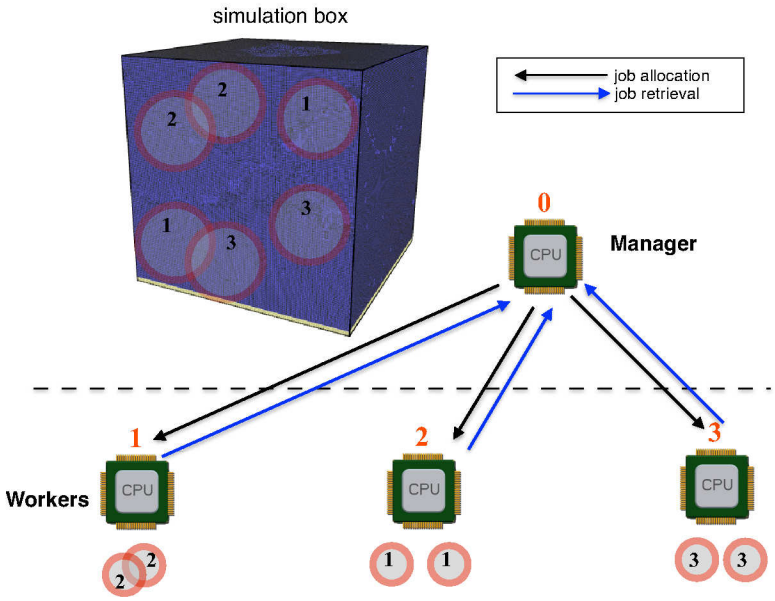

Figure 2: Example of the general manager-worker parallelization scheme showing the simulation domain administered by the manager process (CPU-0). Different spherical volumes shown refers to the local subdomains with randomly chosen C and virtual atoms as centers, and the number corresponds to the MC step. Information about these spherical volumes (i.e., jobs) are communicated by manager to worker processes (CPU - 1 to 3).

In summary, the general manager-worker approach using speculative scheme was realized to perform generic, ensemble-independent, unbiased, and uniform sampling with reasonable efficiency. A more detailed description of the coupling scheme for solute segregation and this parallelization approach is provided elsewhere ([7, 22, 23]).

3.2. Distributed manager-worker

Theoretical limitations for the general manager-worker scheme are expected for two main reasons: (1) the scalability of the scheme, which is limited if the manager process gets overloaded as the number of workers increases, and (2) the

system size, because of the storage of global information in the memory of the manager process. To solve these issues, a distributed manager-worker approach with an asynchronous communication scheme is implemented (also using C++ with the MPI library, as the general algorithm), and according to the model shown in Figure 3. The basis of this distributed approach is the use of a domain decomposition approach, so that the simulated system is divided into contiguous subdivisions (domains) assigned to different manager processes (i.e., one domain per manager). All trial moves generated by each manager are executed by its associated worker process. The underlying system implementation is defined by a generic simulation framework based on the generic definition of domains, blocks, and cells [24]: a domain is defined spatially as a 3-dimensional mesh of equally sized blocks, each block is a 3-dimensional mesh of equal cells, and each cell contains a set of atom information required for its assigned volume. By definition, the minimum volume of a cell in the system is $L_{Fe} \times L_{Fe} \times L_{Fe}$, where L_{Fe} represents the lattice constant of Fe (2.8665 Å).

3.2.1. Trial move processing

As a consequence of the domain decomposition, each manager has only partial system information, and the execution of trial moves has to be coordinated among the managers. Thus, each manager process is initially assigned a balanced number of MC trial moves and uses a coordination protocol with the rest of managers to select the target and sample atoms for every trial move. The target C atom is always chosen locally by random selection inside the set of atoms assigned to each manager. On the other hand, the sample virtual atom can be chosen in two ways: either in the proximity of the target (i.e., inside a sphere centered on the target atom using a given radius) or anywhere in the system, thus implementing local or fully non-local trial moves. An example of trial move selection is illustrated in Figure 4.

The selection of a virtual atom in an area outside the local domain is done by obtaining a random cell within the system limits and then choosing a random virtual atom inside that cell. The high density of virtual atoms, which

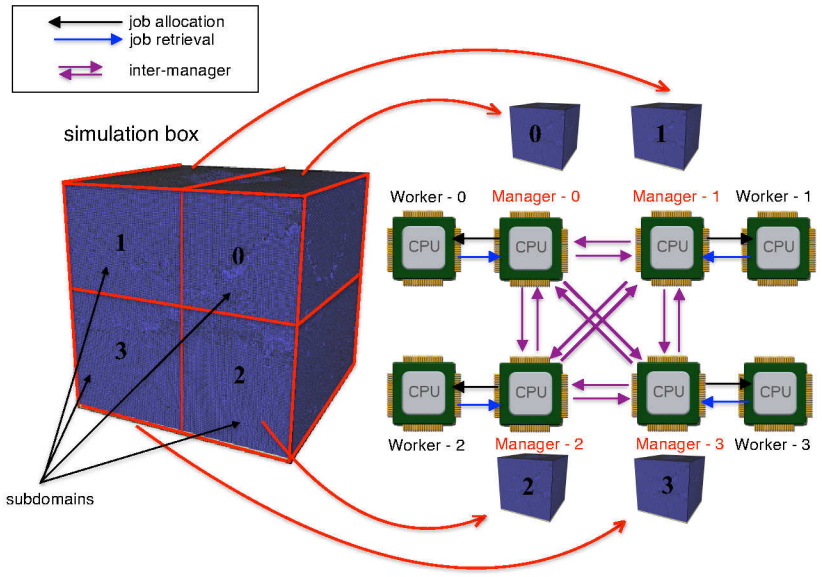

Figure 3: Example of the distributed manager-worker parallelization scheme. The whole computational domain is partitioned and assigned to n manager processes (0-3) as shown. Each manager assigns trial moves to only one associated worker process.

are defined in octahedral sites of the Fe structure with low C concentration, guarantees a very high chance of finding a virtual atom within the size of a cell in the bulk system, even considering the minimum cell size (lattice constant of Fe). Therefore, for any given trial move, the local manager sends the locations of the selected target and the desired virtual atom to all other managers, so that it can obtain the necessary information about atoms in remote domains to build the required spheres. Any remote manager may notify a conflict if the spheres are overlapping with their own local trial move, and also upon selection of a sample cell that includes no valid virtual atom. If the trial move is valid for a remote manager, it returns an accepted tag and the partial system information required by the local manager to build the spheres.

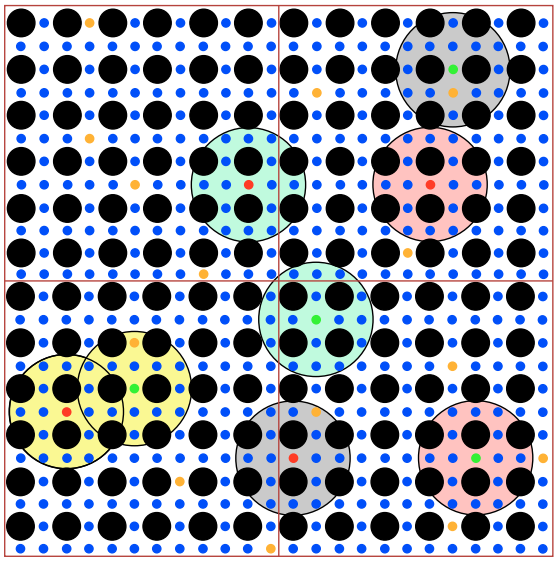

Figure 4: Simplified example of distributed manager worker with four managers and four trial moves associated with them. The Fe atoms are indicated by large dotted black circles, whereas C and virtual atoms are represented by small orange and blue circles, respectively. The target and sample atoms for each given trial move are located in the center of the spheres in red and green, respectively. Note that the local trial move to each manager is the one whose target atom is located in the local domain: sample atoms may be located anywhere else in the system.

After this step, the local manager creates its own part of the spheres, packs the information together and sends it to its associated worker process, alongside with the old value of the energy of the sphere. The worker executes the local MD routine for the relaxation of the spheres, computes the new energy and decides if the trial move should be accepted. The reason to do the acceptance evaluation on the worker is to minimize communication from worker to the manager: apart from the necessary statistical information about the executed trial move, the worker only sends the full information of the spheres to the manager if the trial move is accepted. When the manager receives the information about the trial move, it checks the status (accepted/rejected) and updates its local domain when necessary. To avoid additional messages, the result data from remote managers is stored in a send buffer until the local manager requires to establish a new communication with them: that is, the result of an executed trial move may be sent as an attachment to another message notifying, e.g., a new trial move or a finished manager. As a result of this, a high acceptance ratio of trial moves may only imply more communication of modified atom positions and a local domain update on the managers, but even this overhead is negligible compared to the execution of trial moves performed by the workers.

3.2.2. Load balancing

The use of a load balancing algorithm in the distributed manager-worker approach is essential for the correct behavior of the algorithm. As the C atoms in each domain are taken as targets for the execution of all the trial moves that correspond to each manager-worker pair, it is necessary to keep a good amount of choices for target atoms in the domain. Moreover, a similar number of C and virtual atoms per manager is required to ensure an almost equal chance of selection for any of these elements in a trial move at any time of the simulation. As a consequence of that, load balancing also ensures that representative statistical information is obtained from the MC routine. The distributed manager-worker algorithm implements two types of load balancing: (1) adaptive algorithm with non-orthogonal domains, and (2) atom decomposition procedure. A general

illustrative example of both routines can be found in Figure 5.

On the one hand, the adaptive load balancing algorithm [25] is integrated in the functionality of the framework that gives support to the domain decomposition of the system [24]. Starting from an orthogonal definition of domains, this algorithm moves the vertices of each domain according to a driving force, which is calculated using a system-dependent workload function. The workload function is implemented using heuristic information that can estimate the workload associated with each domain. In the present case, the heuristic information considered is the number of C atoms in each domain. Together with the coordinates of the C atoms, the load balancing algorithm can define a center of gravity for each domain and then move the corners accordingly to obtain a better workload distribution. The corners that define the limits of the systems are not moved to preserve the correct bounding box. Therefore, when the C concentration on one domain goes below a threshold (generally defined as 60% of the ideal distribution of C defined as n_C/n_{dom}, where n_C is the total number of C atoms in the system, and n_{dom} is the number of domains) in one of the domains, the load balancing routine is activated for the whole system and the domain corners are moved until a more even distribution of C is achieved (at least 80% of the ideal distribution of C in all domains).

On the other hand, the load balancing routine based on atom decomposition distributes C atoms in the system between domains separately from the rest of atoms in the system. First, this algorithm defines the initial orthogonal domains according to system size, but the domain structure (blocks and cells) is only filled with the data of Fe and virtual atoms, whereas all the data related to C atoms are stored in a separate array. If the execution of many trial moves

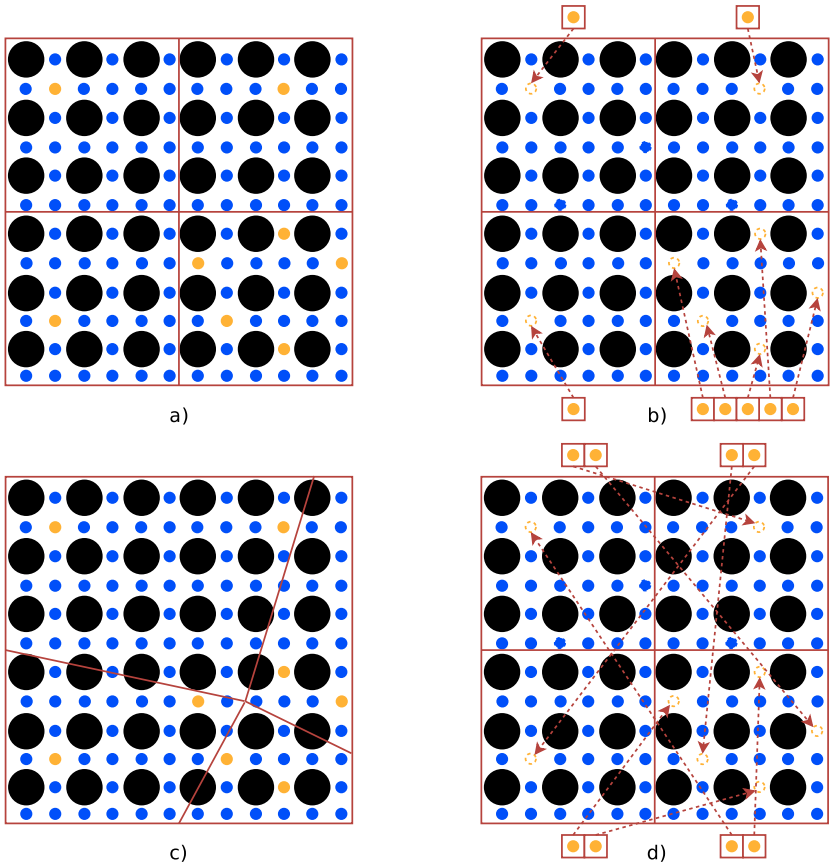

Figure 5: Simplified example of adaptive load balancing (left-hand side) vs. atom decomposition (right-hand side). The graphs *a)* and *b)* (upper part) represent the same initial system distributed in four orthogonal domains, and the graphs *c)* and *d)* (lower part) illustrate two possible scenarios after executing both load balancing routines on the initial systems. The C atoms in the atom decomposition scheme are stored separately in different arrays for each domain (drawn at the top and the bottom of the corresponding graphs), and the dotted arrows indicate which is the atom the correct position of each C atom in the associated domain.

causes the C concentration to drop, the arrays of C atoms for every domain are gathered by one root manager (defined as process 0), shuffled and distributed back to the managers. The domain structure that contains the rest of atoms (Fe and virtual atoms) is also refreshed when the load balancing routine is activated.

The best feature of the adaptive load balancing algorithm is its high flexibility while preserving the neighborhood relations between domains so that the largest data communications for trial move selection can always be defined among nearest neighbor domains. However, this algorithm performs conservative checks for coherency between domains, so that one atom may only move to the nearest neighbor cell between executions of the load balancing routine. Therefore the adaptive algorithm may need to be iterated several times until the domain corners move to the final balanced position, which may also be suboptimal. Here the atom decomposition algorithm has a clear advantage because it reaches the optimal distribution in just one step but at the cost of not preserving the locality of C atoms concerning the local domain (cf. Figure 5). Consequently, a global and potentially large communication among all managers is required to perform the load balancing and also build every trial move sphere with all possible Fe, virtual and C atoms.

Overall, the choice of load balancing algorithm for the MC simulation is mainly related to the number of C atoms and the size of the system, as well as the expected scalability. The fact that the bcc Fe structures for the simulation of Cottrell atmospheres contain a low C concentration implies that the overhead of using the atom decomposition load balancing is relatively small, because the amount of C atoms is almost negligible concerning the number of Fe and virtual atoms. Only when the system size requires the definition of a very large number of domains (potentially in the order of 10^3), the nearest neighbor communications between managers in the adaptive load balancing will be more efficient than a global communication for every trial move. Considering these facts, the selected algorithm for the test cases shown in Section 4 has been the atom decomposition.

3.2.3. Asynchronous processing

As a general summary of the distributed manager-worker functionality, Figure 6 presents the pseudocode of the core routine for the processing of trial moves in each manager. The parallel execution of this routine in every manager is mainly asynchronous, so that the requests for new trial moves and notification of finished managers may be sent and received at any point in time and at any stage of the algorithm. The execution of the load balancing routine is the only part that requires synchronization between managers. After receiving the worker results, the corresponding manager checks if the load balance condition is met, and in that case, the manager sends a trigger to the remote managers in the system and waits for other managers to be ready for load balancing. While waiting for load balancing, a triggered manager will be receiving and processing any notification from remote managers normally: the only difference with the normal status is that no new trial move is generated until the load balance routine has been called. Finally, when all managers are triggered, the load balancing routine is executed, and after it, new trial moves can be generated.

4. Performance results

The manager-worker MC codes (general and distributed) have been tested in the JURECA supercomputer at Jülich Supercomputing Centre (JSC), which represents a general-purpose supercomputer with hybrid shared/distributed memory architecture. Each computing node is composed of two Intel Xeon E5-2680 Haswell CPUs (2×12 cores) and 128 GB DDR4 memory, with the interconnection network Mellanox EDR InfiniBand. The compiler suite used for the MC applications and the internal MD library used for the sphere relaxation of every trial move (the IMD library [20]) is the Intel C compiler suite 18.0.0, with the MPI wrapper `mpicxx` for the parallel MC and `icc` for the serial MD library call. The optimization flags "-Ofast -xHost -fno-alias" have been used in all compilations.

Two different test systems are used for this evaluation. The description and

```
numtrialmoves = N
finished = 0
numactivemanagers = numtotalmanagers
while (finished < numtrialmoves or numactivemanagers > 0) {
  if (load balancing is triggered and
      all other managers are ready) {
    execute synchronous load balancing routine
    if (local manager is active) select new trial move
  }
  wait message from other manager or the worker
    if (message type is WORKER_FINISHED) {
      if (previous trial move is accepted) {
        update sphere and local domain
        pack sphere data to send to remote managers
      }
      increase finished
      if (finished == numtrialmoves) {
        decrement numactivemanagers
        send MANAGER_FINISHED and packed data to all managers
        continue to next iteration of the outer loop
      }
      if (number of local targets is too low) {
        activate local trigger of load balancing
        send LOAD_BALANCING and packed data to all managers
        continue to next iteration of the outer loop
      }
      new_trial_move_OK = false
      while (not new_trial_move_OK) {
        select new trial move
        if (no conflicts reported from remote managers) {
          new_trial_move_OK = true
        }
      }
      send NEW_TRIAL_MOVE and packed data to all managers
    }
    else if (message type is NEW_TRIAL_MOVE) {
      if (data from previous remote trial move is sent) {
        update local domain
      }
      process remote trial move
      if (conflict with local trial move) {
        notify CONFLICT to the sender
      }
      else {
        if (remote trial move covers local domain) {
          create sphere of trial move with local data
        }
        notify NO_CONFLICT with sphere data if required
      }
    }
    else if (message type is FINISHED) {
      if (data from previous remote trial move is sent) {
        update local domain
      }
      decrement numactivemanagers
    }
    else if (message type is LOAD_BALANCING) {
      if (data from previous remote trial move is sent) {
        update local domain
      }
      activate local trigger of load balancing
    }
  }
}
```

Figure 6: Manager pseudocode structure of the asynchronous scheduling algorithm for trial moves using synchronous load balancing.

generation procedure of the systems is as follows:

- **Cylindrical disc:** A defect-free single crystalline bcc ferrite box with the 'virtual atoms' at all the octahedral sites is initially created and relaxed using MD [20] to achieve the ground state structure. The box dimension read 48 nm×48 nm×7 nm along the X [$\bar{1}$ 2 $\bar{1}$] ,Y [$\bar{1}$ 0 1] and Z [1 1 1] direction.

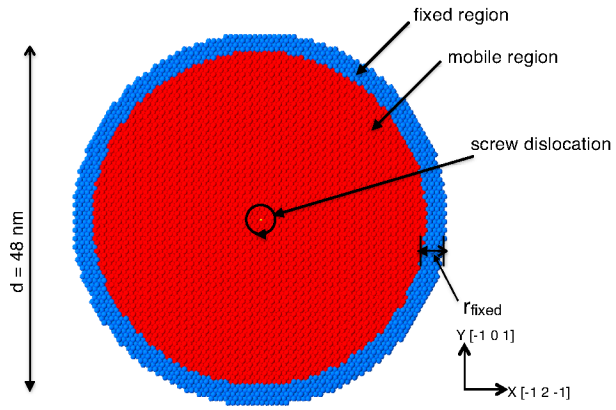

Figure 7: Disc-type simulation box with a radius $R = 24$ nm containing a screw dislocation at the center. This cylindrical disc has two special regions: mobile and fixed (restricted mobility). Fixed boundary conditions were imposed along the X and Y direction, with periodicity applied along the Z direction.

In a follow-up step, a screw dislocation is introduced in the center of the box using the linear elastic solutions of a displacement field followed by a structure relaxation to equilibrate the system. The introduced dislocation line passes along the Z [1 1 1] direction. As a result, the virtual atoms near the dislocation core rearrange to new energetically favorable sites from their initial positions. Then, a cylindrical disc with radius $R = 24$ nm is extracted from the box and a fixed region is introduced as shown in

Figure 7. Furthermore, a suitable background concentration of C atoms (0.01 wt.%) is distributed homogeneously by changing some of the random virtual atoms into C atoms. The atomic composition consists of 1.04 $\times 10^6$ Fe atoms, 2.9 $\times 10^6$ virtual atoms and 500 C atoms.

- **Nanoindentation:** A defect-free sample cube of volume 50 nm^3 is initially created with the crystallographic orientations, X [1 0 0], Y [0 1 0] and Z [0 0 1] (cf. Figure 1). The created sample contained bcc-Fe atoms and virtual atoms at all the octahedral sites. Additionally, a background concentration of C atoms is introduced at uniformly sampled random octahedral sites, representative for a homogeneous distribution. As the introduced C atoms create some tetragonal lattice distortion, the crystal structure is relaxed at temperature $T = 0K$ using Molecular Statics to minimize the system energy. In the following step, a nanoindentation is performed using MD [20] on a relaxed sample cube with a spherical indenter of radius 8 nm. Periodic boundary condition are assigned along the X [1 0 0] and Y [0 1 0] direction. In the Z [0 0 1] direction, the free boundary condition is applied on the top surface of the sample, and several atomic layers of thickness 1 nm from the bottom surface is fixed. The spherical indenter drives into the defect free material in -Z direction at a velocity of 20 m/s for a total duration of 220 ps. Below the contact point of the indenter, dislocations are nucleated during the early phase of deformation of the surface. When the indenter is pressed down further, new dislocations are generated by nucleation and multiplication mechanism. The existing dislocations propagate and form a complex dislocation network undergoing further plastic deformation. The simulation box contains 10.72 $\times 10^6$ Fe atoms, 43.6 $\times 10^6$ virtual atoms and 2000 C atoms.

In general, the measurements of trial move executions present some significant variability because of the use of real defect structures as simulation systems. The most time consuming part of the MC routine is the relaxation of the spherical regions associated to each trial move, which may significantly vary

depending on the part of the system where these spheres are chosen. Therefore, in order to provide statistically relevant results, a preliminary analysis of relaxation times for many thousands of trial moves has been performed on the target systems. This analysis indicated that, for all the studied cases, at least 90% of all trial moves in any of the systems used 700 or fewer relaxation steps to get the spherical regions converged, regardless of the system used. Considering this fact, and with the aim of obtaining reliable execution times by minimizing the impact of very long relaxations, the IMD library has been configured to allow a maximum of 1000 relaxation steps per trial move. In this evaluation, the results from Figures 8 and 9 have been obtained according to this principle.

The tests accomplished in this performance evaluation have been classified in two types of comparison: (1) general vs. distributed manager-worker approaches with different systems and amounts of processes, and (2) use of local vs. non-local trial moves.

4.1. Comparison of manager-worker algorithms

The number of worker processes is set as the reference value for this performance comparison. Therefore, the total number of MPI processes (sum of managers and workers) used for the simulations varies depending on the number of managers used by the algorithm. As an example, a performance result with N workers means that the corresponding test case with the general manager-worker has $N + 1$ processes, whereas the same test case with the distributed manager-worker has $2N$ processes). All test cases in this evaluation have been executed using up to 16 processes (managers + workers) per computing node. Consequently, the executions with the distributed manager-worker for 16 or more workers occupy twice the amount of computing nodes as the general manager-worker (in tests up to 8 workers, both algorithms only occupy one computing node).

Figure 8 presents the execution times of the general and distributed manager-worker approaches for the simulation of up to 4000 trial moves using the cylindrical disc system and with a different number of worker processes: one graph

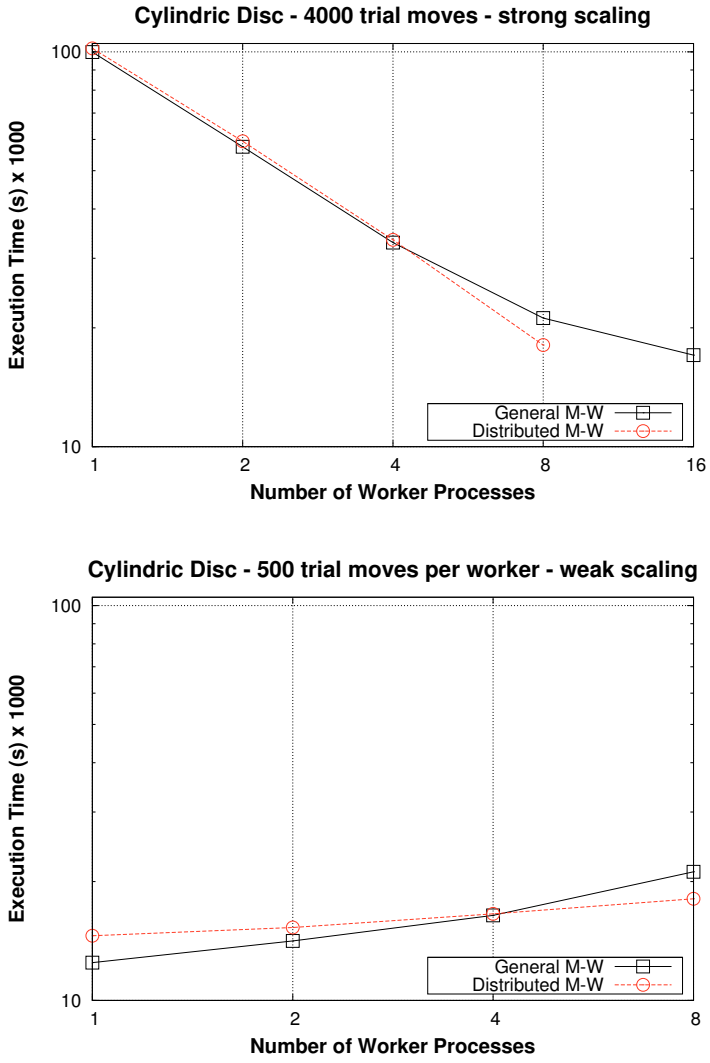

Figure 8: Strong and weak scaling tests for general and distributed manager-worker algorithms with the cylindrical disc.

shows strong scaling results, whereas the other includes the weak scaling results for the execution of 500 trial moves per process. The performance measurements indicate that the general manager-worker approach is performing more efficiently for up to 4 worker processes, but the results with 8 workers are better for the distributed manager-worker. Nevertheless, the latter algorithm cannot get to execute this system with a larger amount of workers because of the limitation of not allowing overlapping trial moves among different workers (i.e., no speculative execution is considered): it is not possible to guarantee that there is enough space to fit, e.g., 16 concurrent trial moves executions inside the cylindrical disc considering the random distribution of C atoms and the random selection of samples. This overlap limitation has been introduced for simplicity in the implementation of the distributed domain update for the different managers, and it is also one reason why the general manager-worker performs better, apart from the fact that the system size is quite manageable for a single manager with a couple of workers.

Despite the restriction in the number of workers, it is important to note that the distributed manager-worker obtains a performance result for 8 workers that is comparable to the general manager-worker with 16 workers (i.e., in two test cases where both algorithms have the very similar amount of processes: 16 and 17, respectively). This indicates that a large amount of workers does not directly imply higher efficiency because of the spatial conflicts between trial moves, especially in the case when many conflicts and even chains of dependencies between trial moves need to be solved: a conflict between trial moves that are accepted may imply that one or more trial moves have to be discarded and recalculated, as mentioned in Section 3.1.

Figure 9 shows the execution times using the nanoindentation system for both general and distributed manager-worker on 800 trial moves. The results up to 4 workers are quite similar for the two tested algorithms, but the distributed manager-worker outperforms the general version when 8 or more workers are used. This situation is mainly due to the system size: the only manager of the general algorithm has to perform random accesses to gather sphere information

in a quite large system, and this overhead adds up to the bottleneck of accessing the same manager by every worker, as in the previous case of Figure 8. This effect of the system size is by definition not present in the distributed algorithm because of the domain decomposition and communications, and the results in this graph show that the overhead of communications is still low compared to the management workload. Considering the worst-case scenario when all C atoms are shuffled after load balancing and no local information from the manager can be used for the execution of a local trial move, the maximum communication size between two managers is the size of two spheres, and this is never dependent on system size, but on the system type (Fe-C). Additionally, the distributed algorithm uses the underlying cell-based framework with tailored block definitions according to the size of a sphere [24], which also helps to obtain a more efficient sphere construction. As a result, the tests confirm the potential benefits of the distributed manager-worker approach for large systems.

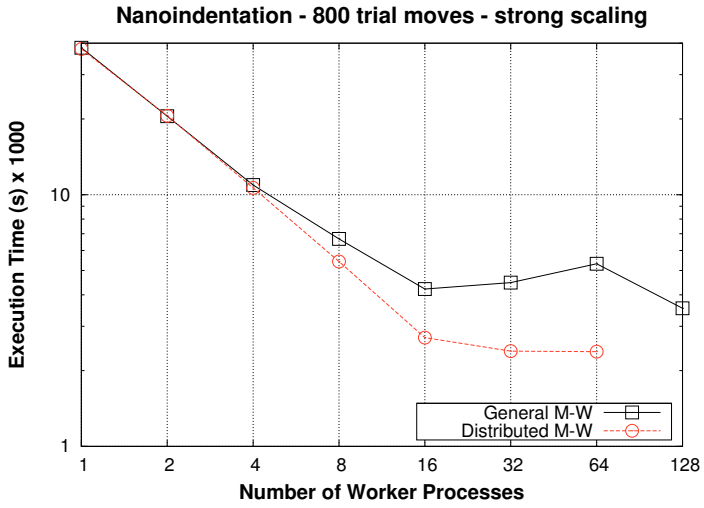

Figure 9: Strong scaling test for general and distributed manager-worker algorithms with the nanoindentation.

It is especially remarkable that the result of the distributed manager-worker with 16 workers is better even than the test cases with the general manager-

worker with 128 workers, and also that the results for 64 workers with the general algorithm are worse than with 32 and 128 workers. To clarify this behavior, Table 1 compares the execution time results shown in Figure 9 with the same tests obtained with just one less worker process (second and fourth columns, starting from the left). The third and fifth columns present the number of conflicts between trial moves associated with the previous tests, respectively. The comparative results indicate that the use of an odd number of MPI processes in the first column (65 processes) represents a drawback for performance when using more than two computing nodes with the general manager-worker approach (i.e., for 64 or more processes), because of an even number of processes allows more efficient matching with the underlying computer architecture, thus better intra-node performance. Analogously to this, when up to 32 workers are used, the performance achieved with $n - 1$ workers causes a performance loss concerning n workers.

Another factor that influences the performance variation is related to the number of conflicts for the general manager-worker algorithm, because of discarded trial move executions: when the number of conflicts goes to thousands, there is a much higher chance of recalculation of trial moves and additional inter-process communications than with the hundreds of conflicts.

The main drawback here is that the influence of these facts is very hard to measure or predict quantitatively because of the inherently random nature of the trial moves and the scheduling system of the manager, i.e., the number of conflicts only implies a proportional *probability of influence* in the execution time. Thus, it may effectively appear as the number of conflicts increases in orders of magnitude. In the case of the distributed manager-worker, conflicts are directly avoided because of no conflicting trial moves are allowed to be executed in parallel, so these problems do not exist.

In general, for the simulation of relatively small to medium sized systems, the general manager-worker gets good performance and allows scaling up to several workers. The possible execution of overlapping trial moves controlled centrally by one manager also represents the advantage of this approach over the domain

n	$1M + nW$	$Num.Conflicts$	$1M + (n-1)W$	$Num.Conflicts$
16	4219.94	182	4533.43	171
32	4473.45	1042	5185.41	886
64	5314.83	1397	4531.45	1910
128	3538.71	1854	3374.60	1652

Table 1: Performance details for the nanoindentation simulation with the general manager-worker approach. The execution times according to the number of manager (M) and worker (W) processes is shown in columns 1 and 3, and columns 2 and 4 contain their respective numbers of conflicts between trial moves.

decomposition. Nevertheless, for the case of the cylindrical disc, the efficiency decreases as both the number of workers and system size grow. The results of the distributed manager-worker imply the need for a good ratio between the volume covered by the simultaneous trial move computations (which is related to the number of managers and workers) and the total system size. Regardless of the system size, just the use of a large number of simultaneous managers can cause a randomly large amount of conflicts, which may imply a significant performance drop. This fact is less striking when the overlap between different trial moves is not allowed, because it may be possible to generate another trial move in a different part of a large system. Overall, it is clearly shown that the scalability of the distributed manager-worker algorithm comes at the cost of using a larger number of manager processes, but each of them performs less intense management tasks than the single manager in the general approach: consequently, the tradeoff between used cores and algorithm performance is positive.

4.2. Comparison of local and non-local trial moves

The parallel computation of trial moves is also tested using the definition of a trial radius, which represents a limitation in the distance of the two spheres that define each trial move. Therefore, this new parameter allows for C atoms to be swapped with virtual atoms within a given radius. The goal of this comparison

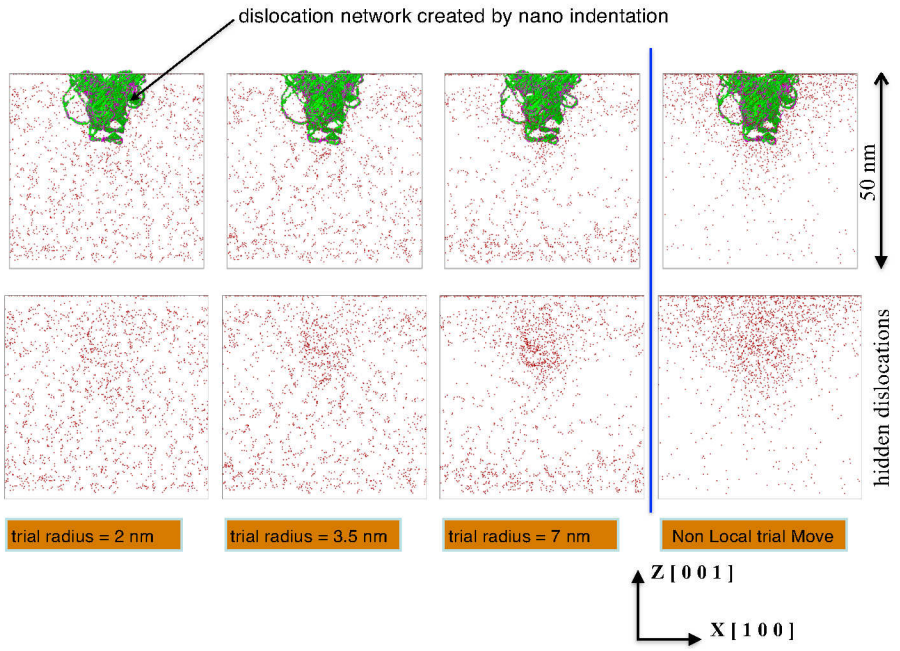

Figure 10: Simulated results of C (*red* colored atoms) segregation in the nanoindented sample after 170000 MC steps. The general manager-worker approach is used to observe the influence of different trial radii (i.e., local trial move) in comparison with the non-local trial move. Fe and virtual atoms are hidden for a clear visualization.

is to show the possibilities of optimization of the MC process at different stages of the simulation.

On the one hand, Figure 10 presents the qualitative analysis of the C segregation process in a nanoindented system and with random C distribution with the general manager-worker approach, showing the configurations achieved after 170000 MC steps using different values for the trial radius, namely 2 nm, 3.5 nm, 7 nm and non-local (infinitely large) radius. The different images highlight the C atoms using red dots and the dislocation network using green color, whereas the Fe atoms that belong to a perfect bcc crystal structure are not shown for the purpose of clarity, and all simulations are started with the same initial con-

figuration, containing 0.01 wt.% of randomly distributed C atoms. Here it is possible to observe a different C distribution for the MC algorithm according to the underlying trial radius. In case of a non-local trial radius, the configuration space gets explored faster, and C atoms initially located far away from the dislocation (lower half of the crystal structure) are able to segregate to the upper half, which contains the defect region with higher lattice distortion. Moreover, it is observed than an increase in the trial radius leads to a higher C concentration near the core of the dislocation networks and the indented surface, where major plastic deformation occurred. However, the simulation with smallest trial radius (2 nm) shows a nearly homogeneous distribution of C atoms after the same number of MC steps: this fact indicates that it is necessary to perform a larger number of trial moves for the case of a small trial radius to move a C atom over a large distance (e.g., if it was located initially at the bottom of the system), whereas this movement could have been achieved with a single accepted trial move if no trial radius had been defined.

On the other hand, Figure 11 illustrates the evolution of the total energy of the nanoindentation using two types of system configurations: (1) a random C distribution, corresponding to the initial configuration, and (2) a non-homogeneous distribution of C atoms, corresponding to an MC simulation with 170000 non-local trial moves. In the latter case, the system has already achieved a state in which the C atoms are getting closer to their optimal distribution for the minimization of the total system energy. Starting from these configurations, 6000 additional non-local trial moves in the one case and with a trial radius of 3.5 nm in the other case have been executed. In the system with random C distribution, the energy reduces quickly, as the atoms are far from the optimal energy and a large number of trial moves is likely to reduce the system energy.

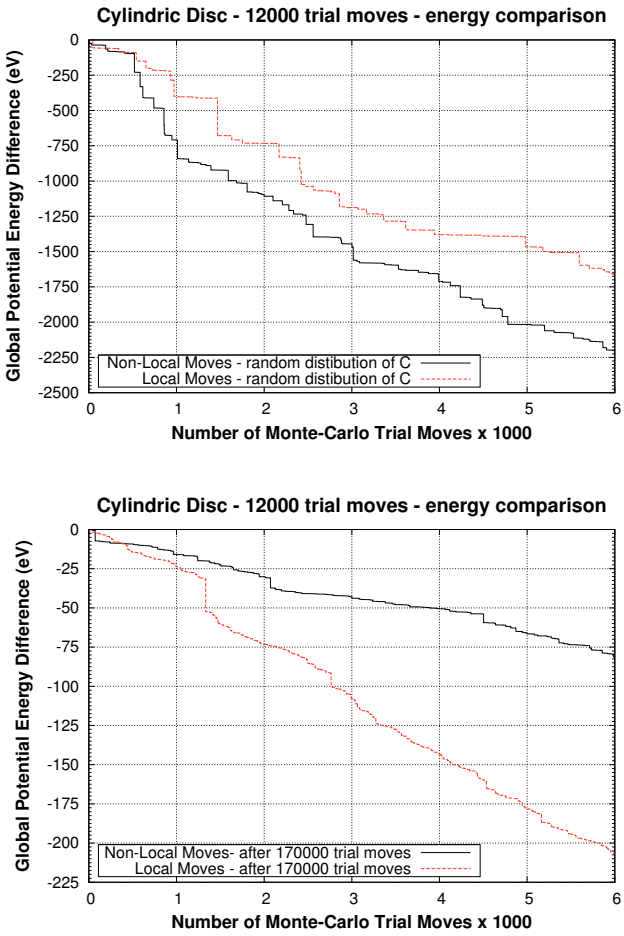

Figure 11: Potential energy variation for the nanoindentation system with the initial random distribution of C and the same nanoindentation after a previous execution of 170000 trial moves. On top of both system configurations, 6000 trial moves are executed and the global energy variation of the system is measured.

This convergence is faster for the non-local moves, which is coherent with the results shown in Figure 10. Nevertheless, the execution of 6000 trial moves in the non-homogeneous configuration shows a highly more optimal energy minimization when the trial radius is finite, which is due to the location of the C atoms. As many C atoms have already been swapped to a position close to the dislocation network, it is statistically much more likely that these atoms find an energetically more preferable position in the local neighborhood than in a randomly chosen part of the system. It is obvious that, in this case, the algorithm has a remarkably lower relaxation time to reach the minimum for the non-local case.

5. Conclusions and future work

This work has presented the comparison of different algorithms for the simulation of Monte Carlo trial moves within a coupling scheme that models C segregation in ferritic structures by swapping the location of C atoms using virtual atoms. The high computational demand of the simulation requires the use of a parallelization scheme based on a distributed model, which has been accomplished using MPI following two approaches: (1) a general manager-worker, with one manager process and many associated worker processes, and (2) a distributed manager-worker, which has many managers and each of them with only has one associated worker. The performance of both algorithms is studied using two main structures with defects (a cylindrical disc and nanoindentation) using up to more than a hundred MPI processes and different trial radii for the range definition of trial moves.

The performance analysis has presented the quantitative comparison between the different parallel algorithms with respect to the size of the system and the number of managers and workers. The results indicate that, for small systems, the use of one manager and a few associated worker processes is beneficial, in such a way that the manager can have faster access to the global knowledge of the system and define a simple and efficient communication pro-

tocol for the parallel execution of trial moves. Nevertheless, the simulation of C segregation on large systems requires a more scalable approach in the aspects of memory and protocol management, so that an increase in the number of workers can be managed efficiently. In this aspect, the tests with the nanoindentation have shown that the simulation does benefit from the use of a domain definition using blocks and cells with sizes adapted to the properties of the system. Moreover, the statistical efficiency of the Monte Carlo procedure has shown to be optimal by choosing random C and virtual atoms for trial move executions on an initial system with random C distribution, but after a large number of executed trial moves, the efficiency increases significantly with the definition of a maximum radius for the distance between the selected atoms in every trial move, i.e., performing local trial moves.

Further developments in this research will cover the implementation of a fully asynchronous scheme for the distributed manager-worker, alongside with a suitable work sharing/stealing routine to change the number of trial moves assigned during runtime or a distributed protocol for the job queue, either using a tailor-made algorithm or additional libraries, such as ADLB [26]. Additionally, the definition of a trigger to switch on a trial radius should be implemented by runtime analysis of statistical information from previous trial moves during a long simulation, and the effective communication between subsets of managers will take the performance of the distributed algorithm to its peak. Therefore, the combination of general and distributed manager-worker algorithms enhanced with runtime information about convergence may represent an optimal approach for solute segregation modeling using the MD-MC coupling method.

6. Acknowledgments

The authors gratefully acknowledge the funding from Deutsche Forschungs Gemeinschaft (DFG) - BE 5360/1-1 and ThyssenKrupp Europe. They also gratefully acknowledge the computing time granted through JARA-HPC on the supercomputer JURECA at Forschungszentrum Jülich.

7. Data availability

The raw/processed data required to reproduce these findings cannot be shared at this time as the data also forms part of an ongoing study.

References

[1] M. A. Meyers, K. K. Chawla, Mechanical Behavior of Materials, Cambridge University Press, 2008.

[2] A. Cottrell, B. Bilby, Dislocation theory of yielding and strain ageing of iron, Proceedings of the Physical Society. Section A 62 (1) (1949) 49.

[3] D. Sourav, S. Shiv Brat, M. Omkar Nath, Encyclopedia of Iron, Steel, and Their Alloys, Taylor and Francis, 2016.

[4] C. A. Wert, Diffusion coefficient of C in α-iron, Phys. Rev. 79 (1950) 601–605.

[5] M. P. Allen, D. J. Tildesley, Computer simulation of liquids, Oxford University Press, 1989.

[6] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller, Equation of state calculations by fast computing machines, The Journal of Chemical Physics 21 (6) (1953) 1087–1092.

[7] H. Ganesan, C. Begau, G. Sutmann, MC/MD Coupling for Scale Bridging Simulations of Solute Segregation in Solids: An Application Study, Communications in Computer and Information Science 889 (2017) 112–127.

[8] A. Moatti, J. Goldberg, G. Memmi, Parallel Monte Carlo calculations with many microcomputers, Computer Physics Communications 45 (1-3) (1987) 355–359.

[9] D. M. Jones, J. M. Goodfellow, Parallelization strategies for molecular simulation using the Monte Carlo algorithm, Journal of Computational Chemistry 14 (2) (1993) 127–137.

[10] K. Esselink, L. Loyens, B. Smit, Parallel Monte Carlo simulations, Physical Review E 51 (2) (1995) 1560.

[11] S. Dietrich, I. D. Boyd, Scalar and parallel optimized implementation of the direct simulation Monte Carlo method, Journal of Computational Physics 126 (2) (1996) 328–342.

[12] G. LeBeau, A parallel implementation of the direct simulation Monte Carlo method, Computer Methods in Applied Mechanics and Engineering 174 (3) (1999) 319–337.

[13] B. Sadigh, P. Erhart, A. Stukowski, A. Caro, E. Martinez, L. Zepeda-Ruiz, Scalable parallel Monte Carlo algorithm for atomistic simulations of precipitation in alloys, Physical Review B 85 (18) (2012) 184203.

[14] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comput. Phys. 117 (1) (1995) 1–19.

[15] T. Karakasidis, N. Cholevas, A. Liakopoulos, Parallel short range molecular dynamics simulations on computer clusters: Performance evaluation and modeling, Mathematical and Computer Modelling 42 (7-8) (2005) 783–798.

[16] N. English, J. Tse, Massively parallel molecular dynamics simulation of formation of ice-crystallite precursors in supercooled water: Incipient-nucleation behavior and role of system size, Phys. Rev. E 92 (3) (2015) 032132.

[17] M. S. Daw, M. I. Baskes, Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals, Physical Review B 29 (12) (1984) 6443.

[18] A. Uhlherr, S. Leak, N. Adam, P. Nyberg, M. Doxastakis, V. Mavrantzas, D. Theodorou, Large scale atomistic polymer simulations using Monte Carlo methods for parallel vector processors, Computer Physics Communications 144 (1) (2002) 1–22.

[19] V. I. Yamakov, Parallel grand canonical Monte Carlo (ParaGrandMC) simulation code, https://ntrs.nasa.gov/search.jsp?R=20160007416 (2016).

[20] J. Stadler, R. Mikulla, H.-R. Trebin, IMD: A software package for molecular dynamics studies on parallel computers, International Journal of Modern Physics C 8 (05) (1997) 1131–1140.

[21] Message Passing Interface Forum, MPI: A Message-Passing Interface Standard Version 3.0, `http://mpi-forum.org/mpi-30/` (Sep. 2012).

[22] G. Sutmann, H. Ganesan, C. Begau, Cluster formation in stochastic disk systems, in: AIP Conference Proceedings, Vol. 1863, AIP Publishing, 2017, p. 560089.

[23] H. Ganesan, C. Begau, G. Sutmann, Parallel hybrid Monte Carlo method for segregation of interstitial atoms in the solid state, [in preparation].

[24] C. Teijeiro, H. Ganesan, R. Halver, W. Homberg, G. Sutmann, Towards a flexible cell-based framework for parallel scale-bridging simulations in materials science: a first case study, in: 5th Intl. Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering (PARENG'17), Pécs (Hungary), 2017, paper 24.

[25] C. Begau, G. Sutmann, Adaptive dynamic load balancing with irregular domain decomposition for particle simulations, Computer Physics Communications 190 (2015) 5161.

[26] ADLB: Asynchronous Dynamic Load Balancing, `https://www.cs.mtsu.edu/~rbutler/adlb/` (Accessed May 2017).