Evaluation of the hardware counters for neighbours' selection algorithms in the random cellular automata grain growth model

Mateusz Sitko[1], Kacper Pawlikowski[1], Lukasz Madej[1]
[1] AGH University of Science and Technology, Al. Mickiewicza 30, 30-059 Kraków, Poland
msitko@agh.edu.pl, kpawliko@agh.edu.pl,
lmadej@agh.edu.pl

Keywords: random cellular automata, dynamic recrystallization

1. Introduction

The predictive capabilities of classical finite element models are commonly increased using a digital material representation (DMR) concept [1]. Depending on the approach, the whole procedure can be time-consuming and thus drastically increase the computation time of subsequent model runs. The most straightforward approach for preparing DMRs is processing metallographic images acquired by light or electron microscopy. It is particularly effective for 2D studies because it allows even complex microstructures to be directly mapped to a numerical model based, for example, on finite elements, cellular automata, or Monte Carlo methods. Often, however, such models are insufficient, and the inhomogeneities appearing in 3D space require more sophisticated experimental solutions to provide input data. Therefore, to speed up the process of generating DMR data, various algorithmic solutions have been proposed over the years [2–4]. A frequently used method, especially for 3D investigations, is simulating unconstrained grain growth using various cellular automata (CA) algorithms. However, the results' quality can be drastically limited by the regularized nature of the cellular automaton space. A simple grain growth algorithm based on the random cellular automata (RCA) method can be used as a potential solution to provide high-quality DMR for further analysis [5]. However, the most time-consuming part of that approach, the neighbour-search algorithm, should be optimized from an algorithmic point of view before practical application in full-field analysis. Minimizing computation time is crucial because simulations with large amounts of CA cells need to be done in reasonable simulation time. It involves much more effort during model implementation to optimise the algorithm in terms of execution time, which has to be at an acceptable time. The main goal of the current work is to evaluate the hardware counters to establish the capabilities of each implemented method. The presented research has shown that significantly reducing RCA simulation time with the adequately developed neighbour-search algorithm is possible.

2. Methodology

The generation of a digital material representation model of single-phase Fe-30Ni alloy characterized by stable austenitic microstructure up to room temperature was selected as a case study (Figure 1). Mentioned RCA approach operates in a mesh-free environment within a dynamic cloud of CA cells. Similarly to the classical CA method, the evolution of CA cell states is directly related

The publication is co-financed from the state budget under the programme
of the Minister of Education and Science called "Excellent Science" project no. DNK/SP/548041/2022

Doskonała Nauka
Ministry of Education and Science
Republic of Poland
Republic
of Poland

to the definition of transition rules that have to be applied for each cell, knowing all its neighbors (Figure 1a). Therefore, four neighbor-search algorithms dedicated to the RCA method were benchmarked within the work for the algorithm's complexity and the impact of individual parameters on computational efficiency. The work presents a comparison of the Basic brute-force neighbours search algorithm performance with the developed, more advanced approaches. The first investigated solution is *SortDimension* algorithm [6] which reduces the number of cell comparisons by sorting CA cells according to the selected coordinate axis. The second is a quadtree-based approach (*QuatTree*) [7] adapted to the needs of RCA, and finally, the third is *FixedGrid* algorithm in which cells are grouped into specific subregions.

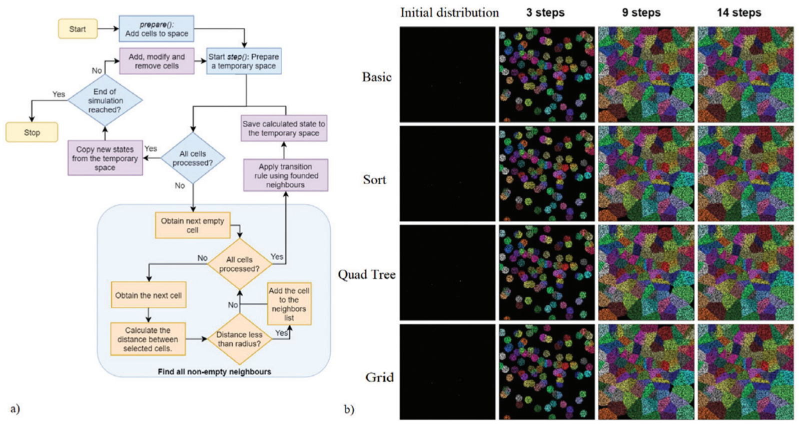
Start
prepare(): Add cells to space
Add, modify and remove cells
Start step(): Prepare a temporary space
No
Yes
End of simulation reached?
Stop
Save calculated state to the temporary space
Copy new states from the temporary space
Yes
All cells processed?
No
Apply transition rule using founded neighbours
Obtain next empty cell
No
All cells processed?
Yes
Obtain the next cell
Add the cell to the neighbors list
No
Calculate the distance between selected cells.
Distance less than radius?
Yes
Find all non-empty neighbours
a)
Initial distribution
3 steps
9 steps
14 steps
Basic
Sort
Quad Tree
Grid
b)

Figure 1. a) general flow of the RCA algorithm, b) RCA simulation results in the form of material microstructure from different neighbor searching algorithms.

3. Results

In order to calculate the average data transfer from RAM, the PAPI_L3_TCM hardware counter and sectional timing were used. The counter mentioned above indicates misses to the L3 cache, which is equivalent to DRAM access. To convert the raw number into bytes, it was multiplied by 64, since, on modern processors, access to memory is done to the entire line, which is most often 64 bytes. The number thus obtained was divided by the time of a given step.

The inspection of branch counters revealed that the smallest number of branch instructions characterized *FixedGridGroup*. This is expected as optimizations incorporated in the algorithms reduced the number of checks. The slowest algorithm has the highest number of conditional branches. Interestingly, the number of accesses to DRAM (misses to L3 cache) was not different among the tested algorithms in the *step()* stage. In the case of *prepare()* stage, the situation is different. The memory on the benchmark machine was DDR4 RAM configured at 2133 MT/s with a bus width of 64bits (single channel). That is around 16GiB/s of theoretical throughput. *FixedGridGroup* algorithm reached a throughput of 7 GiB/s which is only 44% of the theoretical maximal throughput. This stage is not composed only of memory accesses; some minor computations and conditional branches are performed along. This could mean that memory bandwidth is the bottleneck or the algorithm is nearing it.

4. Summary

Various approaches and optimization techniques give the possibility to identify the strong and weak sides of neighbor-searching algorithms. For the simple generation of digital microstructure representation using grain growth algorithm, where cell positions do not change, the method *FixedGridGroup* is fastest, as *step()* stage was optimized at the cost of slowing down *prepare()* stage. In the family of *FixedGrid* algorithms, *FixedSubgrid* showed that mitigating performance penalties for large neighborhoods are possible, but no clear methodology to pick the parameter was found. Further analysis using hardware counters allowed us to gather more detailed information about implemented algorithms and revealed potential bottlenecks. In the *step()* stage, speed was seemingly correlated with the number of conditional branches. In the case of *prepare()* stage, the memory bandwidth is the bottleneck.

References

1. Schmauder S., Schafer I.: *Multiscale Materials Modeling: Approaches to Full Multiscaling*. Walter De Gruyter, 2016.
2. Yang H., Wu C., Li H., Fan X.: *Review on cellular automata simulations of microstructure evolution during metal forming process: Grain coarsening, recrystallization and phase transformation*. Science China Technological Sciences, 54, 2011, 2107–2118.
3. Pal P., Abhishek G.S., Karagadde S.: *A Monte Carlo approach to simulate dendritic microstructures during binary alloy solidification*. Modelling and Simulation in Materials Science and Engineering, 28, 2020, 0–17.
4. Van Nuland T.F.W., Palmeira Belotti L., Van Dommelen J.A.W., Geers M.G.D.: *A novel 3D anisotropic Voronoi microstructure generator with an advanced spatial discretization scheme*. Modelling and Simulation in Materials Science and Engineering, 29, 2021.
5. Czarnecki M., Sitko M., Madej Ł.: *The role of neighborhood density in the random cellular automata model of grain growth*. Computer Methods in Materials Science, 21, 2021, 1–10.
6. Baskett F., Shustek L.J.: *An Algorithm for finding nearest neighbors*. IEEE Transactions on Computers, C–24, 1975, 1000–1006.
7. Samet H.: *The Quadtree and related hierarchical data structures*. ACM Computing Surveys, 16, 1984, 187–260.

Acknowledgements. The financial assistance of the National Science Centre project No. 2019/35/B/ST8/00046 is acknowledged. This research was supported in part by PLGrid Infrastructure.