Distributed particle simulation of flow in complex geometries

A.G. Hoekstra & P.M.A. Sloot
Section Computational Science, University of Amsterdam, The Netherlands

Abstract

Lattice Gas Automata (LGA) and the Lattice Boltzmann method (LBM) are specific Cellular Automata (CA) that can be considered as simplified discrete particle models capable of modeling fluid flow. LGA and LBM inherit the intrinsic parallelism of CA. We discuss our distributed particle simulation of flow, based on a parallel LBM. Efficient parallel execution is possible, provided that (dynamic) load balancing techniques are applied. Next, we present a number of case studies of flow in complex geometries, i.e. flow in porous media and in static mixer reactors.

1 Introduction

Cellular Automata (CA) are discrete, decentralized, and spatially extended systems consisting of large numbers of simple identical components with local connectivity. The rational of cellular automata is not to try to describe a complex system from a global point of view as it is described using for instance differential equations, but modeling this system starting from the elementary dynamics of its interacting parts. The original concept of cellular automata was introduced by von Neumann and Ulam to model biological reproduction and crystal growth respectively. [1, 2] Since then it has been applied to model a wide variety of (complex) systems, in particular physical systems containing many discrete elements with local interactions. [3, 4] Although John von Neumann introduced the cellular automata theory several decades ago, only in recent years it became significant as a method for modeling and simulation of complex systems. This occurred due to the implementation of cellular automata on massively parallel computers. Based on the inherent parallelism of cellular automata, these new architectures made possible the design and development of high-performance software environments. These environments exploit the inherent parallelism of the CA model for efficient simulation of complex systems modeled by a large number of simple elements with local interactions. By means of these environments, cellular automata have been used recently to solve complex problems in many fields of science, engineering, computer science, and economy. In particular parallel cellular automata models are successfully used in fluid dynamics, molecular dynamics, biology, genetics, chemistry, road traffic flow, cryptography, image processing, environmental modeling, and finance [5].

2 Particle simulation of flow

Some 12 years ago theoretical physicists showed that an highly idealized model of a gas, consisting of particles that have a very limited set of velocities and that are confined to a lattice behaves, on average, as an incompressible fluid [6]. This Lattice Gas Automaton (LGA) is a particle model for fluid flow and is a true CA. In this section we shortly introduce LGA and show how it relates to CA. Detailed theory behind LGA is introduced in two recent books. [7, 8]

Consider a hexagonal lattice, as in Fig. 1. Particles live on the links of the lattice, and they can move from node to node. The dynamics of the particles is such that they all move from one node to another. Next, if particles meet on a node, they collide and change direction (see Fig. 1). The collisions are such that they obey the physical constraints of conservation of mass, momentum, and energy.

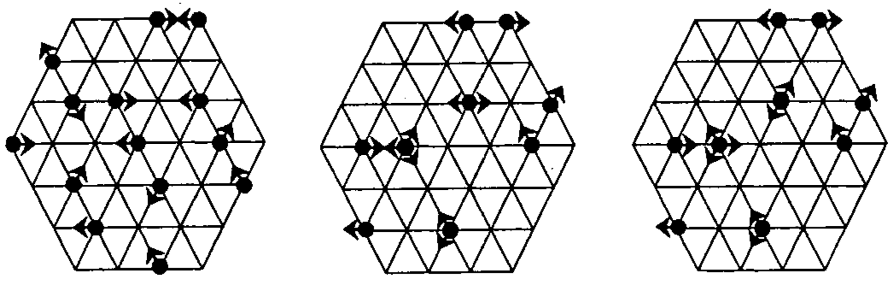

Figure 1: Lattice and particle update mechanism for a LGA. A dot denotes a particle and the arrow its moving direction. From left to right an initial condition, streaming, and collision of particles are shown.

We can formally define a CA rule for such an LGA as follows. Suppose that the state of a cell is determined by b_m surrounding cells. Usually, only the nearest and next-nearest neighbors are considered. For example, on a square lattice with only nearest neighbor interactions b_m = 4, if next-nearest neighbors are also included b_m =8, and on a hexagonal lattice with nearest neighbor interactions b_m = 6. Furthermore, suppose that the state of the cell is a vector **n** of $b = b_m$ bits.

Each element of the state vector is associated with a direction on the CA lattice. For example, in the case of a square grid with only nearest neighbor interactions we may associate the first element of the state vector with the *north* direction, the second with *east*, the third with *south*, and the fourth with *west*. With these definitions we construct the following CA rule (called the LGA rule), which consists of two sequential steps:

1. Each bit in the state vector is moved in its associated direction (so in the example, the bit in element 1 is moved to the neighboring cell in the north) and placed in the state vector of the associated neighboring cell, in the same position (so, the bit in element 1 is moved to element 1 of the state vector in the cell in the north direction). In this step each cell is in fact moving bits from its state vector in all directions, and at the same time is receiving bits from all directions, which are stored into the state vector.
2. Following some deterministic or stochastic procedure, the bits in the state vector are reshuffled. For instance, the state vector (1,0,1,0) is changed to (0,1,0,1).

Maybe very surprisingly, if we assign physical quantities to this CA, enforce physical conservation laws on the bit reshuffling rule of step 2, and use methods from theoretical physics to study the dynamics, we are in fact able to analyze the CA in terms of its average behavior. The average state vector of a cell and the average flow of bits between cells can be calculated. Even better, it turns out, again within the correct physical picture that this CA behaves like a real fluid (such as water) and therefore can be used as a model for hydrodynamics. Furthermore, as the LGA rule is intrinsically local (only nearest and next nearest neighbor interactions) we constructed an inherently parallel particle model for fluid flow.

Associate the bits in the state vector with *particles*; a one-bit codes for the presence of a particle, and a zero bit codes for the absence of a particle. Assume that all particles are equal and have a mass of 1. Step 1 in the LGA-CA is now interpreted as streaming of particles from one cell to another. If we also introduce a length scale, i.e. a *distance* between the cells (usually the distance between nearest neighbors cells is taken as 1) and a time scale, i.e. a *duration* of the streaming (i.e. step 1 in the LGA-CA rule, usually a time step of 1 is assumed), then we are able to define a *velocity* $\mathbf{c}_i$ for each particle in direction i (i.e. the direction associated with the i-th element of the state vector $\mathbf{n}$). Step 1 of the LGA-CA is the streaming of particles with velocity $\mathbf{c}_i$ from one cell to a neighboring cell. Now we may imagine, as the particles meet in a cell that they collide. In this collision the velocity of the particles (i.e. both absolute speed and direction) can be changed. The reshuffling of bits in step 2 of the LGA-CA rule can be interpreted as a collision of particles. In a real physical collision, mass, momentum, and energy are conserved. Therefore, if we formulate the reshuffling such that these three conservation laws are obeyed, we have constructed a true Lattice Gas Automaton, i.e. a gas of particles that can have a small set of discrete velocities $\mathbf{c}_i$, moving in lock-step over the links of a lattice (space is discretized) and that all collide with other particles arriving at a lattice point at the same time.

In the collisions, particles may be sent in other directions, in such a way that the total mass and momentum in a lattice point is conserved.

We can now define for each cell of the LGA-CA a density ρ and momentum $\rho\mathbf{u}$, with $\mathbf{u}$ the velocity of the gas:

$$\begin{aligned} \rho &= \sum_{i=1}^{b} N_i , \\ \rho\mathbf{u} &= \sum_{i=1}^{b} \mathbf{c}_i N_i , \end{aligned} \qquad (1)$$

where $N_i = \langle \mathbf{n}_i \rangle$, i.e. a statistical average of the Boolean variables; N_i should be interpreted as a particle density.

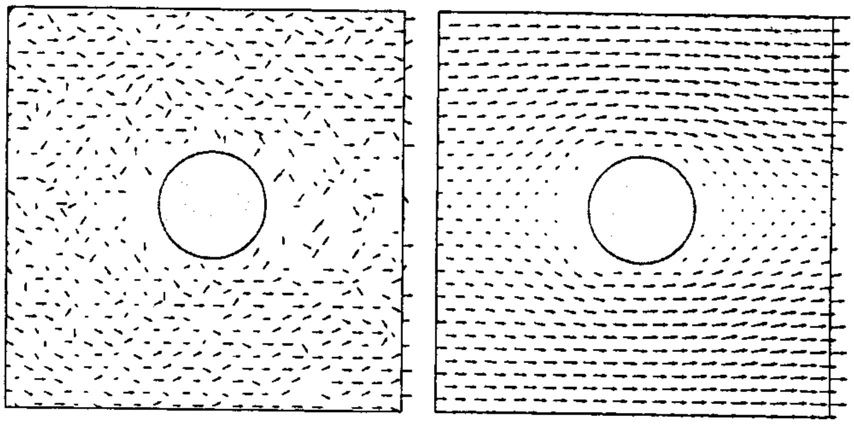

Figure 2: LGA simulation of flow around a cylinder. The arrows are the flow velocities, the length is proportional to the absolute velocity. The simulations were done with FHP-III, on a 32x64 lattice, the cylinder has a diameter of 8 lattice spacings, only a 32x32 portion of the lattice is shown; periodic boundary conditions in all directions are assumed. The left figure shows the result of a single iteration of the LGA, the right figure shows the velocities after averaging over 1000 LGA iterations.

If we let the LGA evaluate and calculate the density and momentum as defined in eqn. (1), these quantities behave just like a real fluid. In Fig. 2 we show an example of an LGA simulation of flow around a cylinder. In left figure we show the results of a single iteration of the LGA, so in fact we have assumed that $N_i = \mathbf{n}_i$. Clearly, the resulting flow field is very noisy. In order to arrive at smooth flow lines one should calculate $N_i = \langle \mathbf{n}_i \rangle$. Because the flow is static, we calculate N_i by averaging the Boolean variables $\mathbf{n}_i$ over a large number of LGA iterations. The resulting flow velocities are shown in the left panel of Fig. 2.

Immediately after the discovery of LGA as a model for hydrodynamics, it was criticized on three points; noisy dynamics, lack of Galilean invariance, and exponential complexity of the collision operator. The noisy dynamics is clearly illustrated in Fig. 2. The lack of Galilean invariance is a somewhat technical matter which results in small differences between the equation for conservation of momentum for LGA and real Navier-Stokes equations, for details see e.g. [8]. Finally, adding more velocities in an LGA leads to increasingly more complex collision operators, exponentially in the number of particles. Therefore, another model, the Lattice Boltzmann Method (LBM), was introduced. This method is reviewed in detail in [9].

The basic idea in LBM is that one should not model the individual particles $\mathbf{n}_i$, but immediately the particle densities N_i. This means that particle densities are streamed from cell to cell, and particle densities collide. This immediately solves the problem of noisy dynamics. However, in a strict sense we no longer have a CA with a Boolean state vector. However, we can view LBM as a generalized CA. It is easy to make LBM Galilean invariant, thus solving the second problem of LGA. Finally, a very simple collision operator is introduced. This so-called BGK collision operator models the collisions as a single-time relaxation towards equilibrium. This L-BGK method is also developed for many other lattices, e.g. in two or three-dimensional cubic lattices with nearest and next nearest neighbor interactions. The LBM and especially the L-BGK has found widespread use in simulations of fluid flow.

3 Parallel Lattice Gas and Lattice Boltzmann Simulations

The local nature of the LGA and LBM interactions allows a very straightforward realization of parallelism. A geometric decomposition of the lattice with only local message passing between the boundaries of the different domains is sufficient to realize an efficient parallel simulation. For instance, we have developed a generic 2-dimensional LGA implementation that is suitable for any multi-species (thermal) LGA [10]. Parallelism was introduced by means of a 1-dimensional, i.e strip-wise, decomposition of the lattice. As long as the grid dimension compared to the number of processors is large enough, this approach results in a very efficient parallel execution.

This LGA system is mainly used for simulations in simple rectangular domains without internal obstacles. However, in a more general case, where the boundaries of the grid have other forms and internal structure (i.e. solid parts where no particles will flow) the simple strip-wise decomposition results in severe load imbalance. In this case, as was shown in [11], a more advanced decomposition scheme, the Orthogonal Recursive Bisection (ORB) method [12], still leads to highly efficient parallel LBM simulations. ORB restores the load balancing again, however at the price of a somewhat more complicated communication pattern between processors. In Fig. 3 we show, for a representative 2D benchmark, the processor load. In a simple slice decomposition, the load on each processor differs a lot. For the ORB the load is almost balanced. This results for this benchmark, in reductions of execution times as large as 40%. For details we refer to [11].

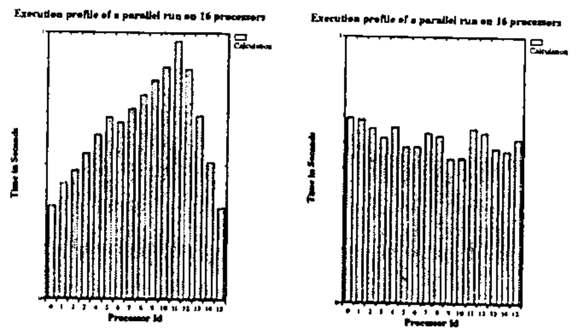

Figure 3: The processor load, on 16 processors, for slice and ORB decomposition of the elutriator chamber benchmark (see [11]). On the x-axis are the processor ID's and on the y-axis the execution time of each processor.

For flow problems with a static geometry the ORB decomposition seems to be most appropriate. However, we are also interested in flow simulations where the geometry dynamically changes during the simulation. Here we refer to e.g. free flow around growing biological objects [13] or bounded flows with dynamically changing boundaries (e.g. blood flow in the heart). In this case an initially well-balanced parallel computation may become highly unbalanced. This may be overcome by a redundant scattered decomposition or by dynamic load balancing. Here we report some preliminary results of the first approach, the latter will be published elsewhere [14].

We consider growth of an aggregate in a three dimensional box. We must simulate flow around the object. The details of the streamlines close to the object determine its local growth. Due to the growth of the aggregate a straightforward decomposition (for example partitioning of the lattice in equal sized slices or boxes) would lead to strong load imbalance. To solve this problem we have tested two strategies to obtain a more equal distribution of the load over the processors:

1. Box decomposition in combination with scattered decomposition.
2. Orthogonal Recursive Bisection (ORB) in combination with scattered decomposition.

The idea is to decompose the grid in a large number of partitions, much larger than the number of available processors, using either a box decomposition or ORB. Next, the partitions are randomly assigned to a processor. An example of a scattered decomposition over 4 processors of an irregular shaped object in a 2D lattice is shown in Fig. 4. In this example the lattice is divided into 100 blocks, where each block is randomly assigned to one of the four processors. Most of the computation is done in the blocks containing exclusively fluid nodes. The scattering of the blocks over the processors leads to a spatial averaging of the load, where decreasing block sizes cause a better load balancing, but also an increasing communication overhead. [15] Especially in simulations in which the shape of the object cannot be predicted, redundant scattered decomposition is an attractive option to improve the load balance in parallel simulations [16, 17].

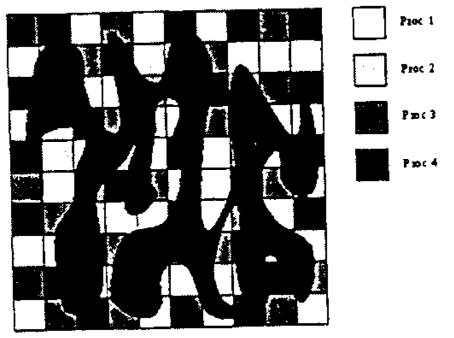

Figure 4: Decomposition of an irregular shaped object in a 2D lattice. In this case 100 blocks are scattered over 4 processors.

We have compared both decomposition strategies by computing the load balancing efficiency:

$$\varepsilon_{load} = \frac{l_{\min}}{l_{\max}}, \tag{2}$$

where $l_{\min}$ is the load of the fastest process and $l_{\max}$ the load of the slowest process. The two decomposition strategies were tested by using two extreme morphologies of the aggregates, a very compact shape and a dendritic shaped aggregate, and by measuring the load balancing efficiency during one iteration of the parallel LBM on 4 processors. The results are shown in Fig. 5. These results indicate that the combination of redundant ORB or box decomposition in combination with a scattered decomposition of partitions on processors may indeed improve load balancing. However, a disadvantage is that although the load balancing efficiencies increase with the number of redundant blocks, the communication overhead also increases. Furthermore, this test was for a single iteration only. We are currently working on testing these methods for real growing objects.

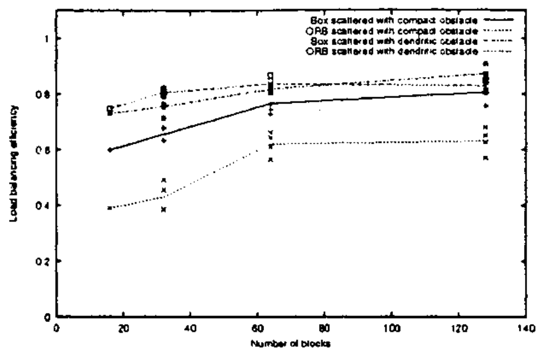

Figure 5: The load balancing efficiency as a function of the total number of redundant blocks scattered on 4 processors, for box and ORB decomposition and for a compact and dendritic aggregate.

4 Cases: flow in complex geometries

We applied our distributed particle simulation environment as described above for a large number of simulations of fluid flow in complex geometries. Here we show two representative examples, that of flow in random fiber networks and of flow in a static mixer reactor.

The first example is flow in a random fiber network, as drawn in Fig. 6. The fiber network is a realistic model of paper, and the question was to obtain the permeability of the network as a function of the volume fraction of fibers. Simulations were performed on 32 nodes of a Cray T3E using our parallel LBM environment described in the previous section. We obtained permeability curves that are in very good agreement with experimental results (see [18]).

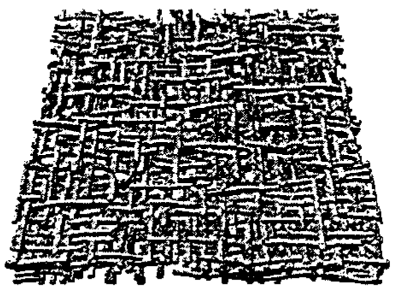

Figure 6 : A random fiber network.

Another impressive example is flow in a Static Mixer Reactor [19]. In such a mixer high viscosity fluids are mixed by letting them flow around a complex arrangement of internal tubes. A typical mixer is shown in Fig. 7. Here, LBM simulations and conventional Finite Element simulations where compared, which agreed very well. The simulation results also agree very well with experimental results. This shows that LBM, which is much easier to parallelize and much easier to extend with more complex modeling compared to Finite Element, (multi-species flow, thermal effects, reactions), is very suitable in real life problems involving complex flow.

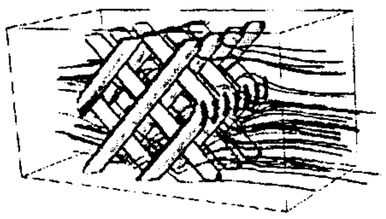

Figure 7 : A Static Mixer Reactor. Flow lines resulting from LBM simulations are also shown.

5 Conclusions

Within the general concept of Cellular Automata we have developed a distributed particle simulation environment for fluid flow. Parallel Lattice Gas Automata and Lattice Boltzmann methods have been realized, and we showed that by carefully taking load balancing into account it is possible to achieve very high parallel efficiencies. By means of two realistic examples of flow in complex geometries the power and potencies of such a distributed particle simulation environment were demonstrated.

References

[1] Ulam, S. Some mathematical problems connected with patterns of growth figures. *Essays on Cellular Automata*, eds. A.W. Burks, Illinois, pp. 1970.
[2] von Neumann, J. *Theory of Self-Reproducing Automata*, Urbana, 1966.
[3] Wolfram, S. *Cellular Automata and Complexity*, Addison-Wesley: 1994.
[4] Manneville, P., Boccara, N., Vichniac, G.Y. & Bidaux, R. (eds.) *Cellular Automata and Modeling of Complex Physical Systems*, Springer Proceedings in Physics: 1989.
[5] Sloot, P.M.A. & Talia, D. (eds.) *Parallel Cellular Automata: Special Issue on Cellular Automata*, Future Generation Computer Systems: 1999.
[6] Frish, U., Hasslacher, B. & Pomeau, Y. Lattice-gas automata for the Navier-Stokes equation. *Phys. Rev. Lett.*, **56**, pp. 1505, 1986.

[7] Chopard, B. & Droz, M. *Cellular Automata Modelling of Physical Systems*, Cambridge University Press: 1998.

[8] Rothman, D.H. & Zaleski, S. *Lattice-Gas Cellular Automata, Simple Models of Complex Hydrodynamics*, Cambridge University Press: Cambridge, 1997.

[9] Chen, S. & Doolen, G.D. Lattice Boltzmann Method for Fluid Flows. *Ann. Rev. Fluid Mech.*, **30**, pp. 329, 1998.

[10] Dubbeldam, D., Hoekstra, A.G. & Sloot, P.M.A. Computational Aspects of Multi-Species Lattice-Gas Automata. *Proceedings of the International Conference HPCN Europe '99*, eds. P.M.A. Sloot, Bubak, M., Hoekstra, A.G. & Hertzberger, L.O., Springer Verlag: Berlin, pp. 339-349, 1999.

[11] Kandhai, D., Koponen, A., Hoekstra, A.G., Kataja, M., Timonen, J. & Sloot, P.M.A. Lattice Boltzmann Hydrodynamics on Parallel Systems. *Comp. Phys. Comm.*, **111**, pp. 14-26, 1998.

[12] Simon, H.D. Partioning of unstructured problems for parallel processing. *Computing Systems in Engeneering*, **2**, pp. 135-148, 1991.

[13] Kaandorp, J.A., Lowe, C., Frenkel, D. & Sloot, P.M.A. The effect of nutrient diffusion and flow on coral morphology. *Phys. Rev. Lett.*, **77**, pp. 2328-2331, 1996.

[14] Schoneveld, A. & de Ronde, J., accepted for publication in Fut. Gen. Comp. Syst., 1999.

[15] de Ronde, J., Schoneveld, A. & Sloot, P.M.A. Load balancing by redundant decomposition and mapping. *High Performance Computing and Networking (HPCN'96)*, eds. H. Liddell, Colbrook, A., Hertzberger, B. & Sloot, P., pp. 555-561, 1996.

[16] Machta, J. & Greenlaw, R. The parallel complexity of growth models. *Journal of Statistical Physics*, **77**, pp. 755-781, 1994.

[17] Machta, J. The computational complexity of pattern formation. *Journal of Statistical Physics*, **70**, pp. 949-967, 1993.

[18] Koponen, A., Kandhai, D., Hellin, E., Alava, M., Hoekstra, A., Kataja, M., Niskanen, K., Sloot, P. & Timonen, J. Permeability of three-dimensional random fiber webs. *Phys. Rev. Lett.*, **80**, pp. 716-719, 1998.

[19] Kandhai, D., Vidal, D., Hoekstra, A., Hoefsloot, H., Iedema, P. & Sloot, P. Lattice-{Boltzmann} and Finite-Element Simulations of Fluid Flow in a {SMRX} Static Mixer. pp. 1999.