Coupling method for building a network of irrigation canals on a distributed computing environment

Mohamed Ben Belgacem, Bastien Chopard, and Andrea Parmigiani

University of Geneva,
{Mohamed.Benbelgacem,Bastien.Chopard,Andrea.Parmigiani}@unige.ch

Abstract. *An optimal management of an irrigation network is important to ensure an efficient water supply and to predict critical situations related to natural hazards. We present a multiscale coupling methodology to simulate numerically an entire irrigation canal over a distributed High Performance Computing (HPC) resource. We decompose the network into several segments that are coupled through junctions. Our coupling strategy, based on the concept of Complex Automata (CxA) and the Multiscale Modeling Language (MML), aims at coupling simple 1D model of canal sections with 3D complex ones. Our goal is to build a numerical model that can be run over a distributed grid infrastructure, thus offering a large amount of computing resources. We illustrate our approach by coupling two canal sections in 1D through a gate.*

Keywords: irrigation canal; lattice Boltzmann models; coupling method; distributed multiscale computing.

1 Introduction

Canals or rivers are important in populated area as they ensure an adequate supply of water for agriculture and are a key component of electricity production or transportation. An optimal management and the control of such resources can be a critical issue for long term planning or to react to natural hazards. The major challenge is to define appropriate actions (e.g. opening and closing gates) that need to be taken to always guarantee an adequate water supply throughout the canal system, whatever the external demands or perturbations can be, and respecting constraints such as water height.

This problem can be addressed through numerical optimization methods. Usually, these methods require the simulation of different scenarios with the canal network subject to different parameters. Therefore numerical methods able to simulate the water flow in a full irrigation system are needed. In order to allow canal operators to respond to real-time events, these methods should compute fast enough, with good accuracy. Due to the size of an irrigation network and the large variation in the flow complexity across different sections, a multi-scale, multi-model computational approach is needed. For instance, some parts of the

system can be described with simple numerical methods, using the shallow water equation, whereas other sections need a 3D, free-surface hydrodynamic model (FS3D) to properly capture the flow properties.

In this paper we propose a methodology to achieve this goal. It follows the line proposed in the MAPPER project [2], where a set of tools, frameworks, and methodologies are developed to provide a standard way to build and run what is defined as *multiscale distributed applications*. For our specific problem, we decompose the overall canal network into segments, called submodels, that will be connected to each other through junctions. There are several reasons that make this decomposition attractive for our simulation: (i) it gives a flexibility to easily reconstruct the canal "on the fly" without the need to recode some submodels; (ii) the different components are reconfigurable and reusable; (iii) it supports multiscale coupling; (iv) the entire simulation can be efficiently carried out across distributed clusters, typically using the large computing resources of the distributed European grid infrastructure.

In this work, we use the Lattice Boltzmann (LB) approach to resolve the equations modeling a canal segment, and the MML language [11, 8] to describe the coupling of the different components of the problem.

The paper is organized as follows. After a brief introduction to the LB method, we describe the coupling strategy and the distributed execution model, as specified in the CxA and MML frameworks. Then, we explain the algorithm that allows the coupling of two 1D canal sections interconnected through a gate, when using the LB method.

2 Lattice Boltzmann method

We briefly present the basic equations of the LB method, which is widely used for computational fluid dynamics. Further details can be found in several papers and textbooks (see for instance [18, 9]).

In the LB method, a fluid is described in terms of q density distribution functions $f_i(\mathbf{x}, t)$, $i = 0, \ldots q-1$, where $\mathbf{x}$ belongs to a Cartesian grid of spacing Δx and t is the discrete time, which varies by steps Δt. From the f_i, the fluid density $\rho(\mathbf{x}, t)$ and the velocity field $\mathbf{u}(\mathbf{x}, t)$ are obtained as $\rho = \sum_i f_i$ and $\rho\mathbf{u} = \sum_i f_i \mathbf{v}_i$. The $\mathbf{v}_i$ are velocity vectors associated with each f_i, chosen such that $\mathbf{x} + \Delta t \mathbf{v}_i$ is also a point of the computational grid. In LB modeling, the lattice is denoted as DdQq where d is the spatial dimension of the Cartesian lattice and q the number of velocity vectors.

In the so-called BGK based LB methods, the density distribution $f_i(\mathbf{x}, t)$ are computed as a relaxation towards prescribed local equilibrium functions f_i^{eq}: $f_i(\mathbf{x} + \Delta t \mathbf{v}_i, t + \Delta t) = f_i(x, t) - \frac{1}{\tau}\left(f_i^{eq}(\mathbf{x}, t) - f_i(\mathbf{x}, t)\right)$, where f^{eq} depends on the physics of the process, and τ is a parameter called the relaxation time. The above equation can also be expressed as a succession of collision and streaming steps:

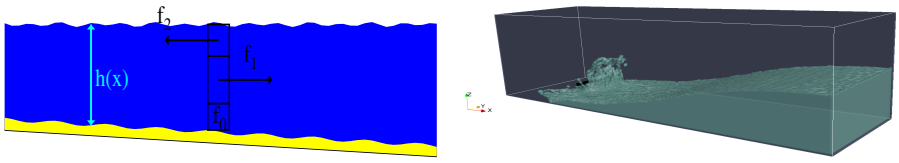
$$\begin{array}{l} \text{Collision:} \quad f_i^{out} = f_i^{in} - \frac{1}{\tau}(f_i^{eq} - f^{in}) \\ \text{Streaming:} \; f_i^{in}(\mathbf{x} + \Delta t \mathbf{v}_i, t + \Delta t) = f_i^{out}(\mathbf{x}, t) \end{array} \tag{1}$$

where $f_i^{in}(\mathbf{x},t) \equiv f_i$ is the density of particles entering, at time t, site $\mathbf{x}$ with velocity $\mathbf{v}_i$; f_i^{out} is the density of particles with velocity $\mathbf{v}_i$ at site $\mathbf{x}$ after collision.

In the shallow water (SW) model, the fluid is described as columns of water with height $h(\mathbf{x},t)$ and velocity $u(\mathbf{x},t)$. For solving the SW equation with a LB approach, $f_i(x,t)$ describes the part of the water column at site $\mathbf{x}$ and time t that travels with velocity v_i.

The one-dimensional case (SW1D) is illustrated in Fig. 1(a). We use a D1Q3 lattice where grid points are separated by a distance Δx. Each lattice site is characterized by three density distributions $f_{i=0..2}(x,t)$, and three velocity vectors $v_{i=\{0..2\}}$: $v_0 = 0$, $v_1 = v$, $v_2 = -v$, with $v{=}\Delta x/\Delta t$. As suggested in Fig. 1(a), the water level h and the velocity u are then defined as $h = \sum_{i=0}^{3} f_i$ and $hu = \sum_{i=0}^{3} v_i f_i$. See for instance [17] for a more detailed description.

The SW1D is sufficient to model a simple canal section where the vertical and perpendicular flow can be neglected. For more complex simulation, a 3D free surface model is needed. An example of 3D free surface model is illustrated in Fig. 1(b), but not further described here (see [5] for more details). In our approach, such a fully resolved 3D simulation is planned to be coupled with the above SW1D model. This 1D-3D coupling will be described in a forthcoming paper. In sect. 4 we rather focus on the specific problem of coupling two 1D canal segments through a gate.



(a) Illustration of the Lattice Boltzmann 1D Shallow water model (SW1D)

(b) Illustration of the 3D free-surface Lattice Boltzmann model (FS3D)

Fig. 1. Two numerical models needed for a multiscale simulation of an irrigation canal.

3 Distributed Multi-scale simulations

One of our objective is to simulate the flow in the "canal de la Bourne" in France. This irrigation network comprises a principal canal (45 km), 4 secondary canals (85 km) and 27 tertiary canals (400 km). After decomposing the entire system into segments, the whole simulation will be performed by coupling all the pieces through junctions. In order to have an acceptable computing time, it is important that the 3D free surface flow simulations are only performed for a limited number of sections, the rest being modeled within 2D or 1D shallow water models.

Such a decomposition produces a simulation involving several submodels (codes) that possibly run with different Δx and Δt and different spatial dimensions. Our goal is to perform the entire simulation on a distributed grid

infrastructure: thus we coin our approach a *distributed, multi-scale simulation* [8]. A specificity of this approach with respect to more standard distributed computing methods is the fact that, here, the submodels are *tightly coupled*: at each iteration, communications may be required and the coupling pattern is represented by a cyclic graph.

When dealing with tightly coupled LB based components, the main concern is to handle inlet and outlet boundaries, which depend on the temporal and spacial resolution, and to synchronize the data exchanges at the borders between canal segments.

3.1 MML component

Simulating multiscale applications remains a challenge for computational sciences since it involves several temporal and spatial scales that requires an interaction of various distributed processes. Cellular automata (CA) and Lattice Boltzmann (LB) models [9] are powerful numerical and theoretical approaches for modeling various complex systems. The concept of Complex Automata (CxA) [6, 14, 15] provide a framework to couple them so as to obtain a multiscale model. A CxA represents a set of "single-scale" LB and/or CA systems, each representing a different physical process at a given scale. They can be coupled together to include all relevant scales and processes into the same simulation. We refer the reader to [15] for more details.

At a practical level, the theoretical CxA framework can be implemented with the MUSCLE API [13, 3], a general coupling software, and the Multiscale Modeling Language (MML) [11]. Both MUSCLE and MML are further developed within the European project MAPPER [2], with the purpose of standardizing the modeling and the execution of several existing multiscale applications over large distributed HPC platforms. MML is an UML-like language which allows scientists to describe a multiscale application, its architectural diagram and its data-flow links in a standard and human-friendly way, using either an XML description (coined XMML) or a graphical tool.

The general formalism offers several components to build a multiscale application. In our case, we use the following ones: submodels, junctions and conduits:

Sub-model: A canal section can be modeled either in one-dimensional shallow water (SW1D) equation or three-dimensional free surface (FS3D) model. The SW1D equation is good to describe long canal sections. We have implemented the SW1D model presented in the previous section, in the Java language. The FS3D model is needed to simulate the water flow around in a complicated geometry, for instance a fully resolved gate, a spillway, or other type of construction for which the shallow water is no longer adequate.

Junctions: A junction is a component that receives data from one or more submodels, performs a computation reflecting the nature of the coupling, and sends updated information back to the corresponding recipient submodels. Junctions can be: gate, spillway, pumping station, or complex structures ...etc. They can be abstracted through a phenomenological equation or actually simulated with a fully detailed flow model.

Conduit: A conduit is a virtual concept used to describe the data-flow links between submodels and junctions.

3.2 Distributed Execution

According to section 3.1, the first step to build a multiscale application consists of specifying the components and their data-flow connections. This can be done with the MML based graphical tools, as depicted in Fig. 2(a). Then the whole diagram can be exported as a virtual experiment package using the GridSpace [10] platform. GridSpace is a platform that enables the end-users to perform transparently a simulation over local or distributed grid and HPC like infrastructures. It supports interfacing with several local resources management systems (LRMS) like PBS [4] and grid middleware like QCGBroker [7].

The second step consists of executing the simulation. This can be done in two ways: local and distributed execution. In case of distributed execution, GridSpace performs the execution of the simulation by creating and scheduling jobs using QCGBroker, which facilitates secure jobs submission and management over distributed clusters. In its simple form, a job is composed of one or more canal segments or junctions running on a cluster node. In case of local simulation, GridSpace utilizes the PBS LRMS to submit jobs on the nodes of the same cluster. When the execution ends, GridSpace retrieves all the simulation results on the front-end node in both execution scenario. It is worthwhile to remind that a job uses the MUSCLE API calls to read its configuration parameters from a CxA description file [14], generated automatically in the virtual experiment package and describing the coupling schema, in order to handle all the send/receive operations among remote jobs. This is shown in the listing 1.1:

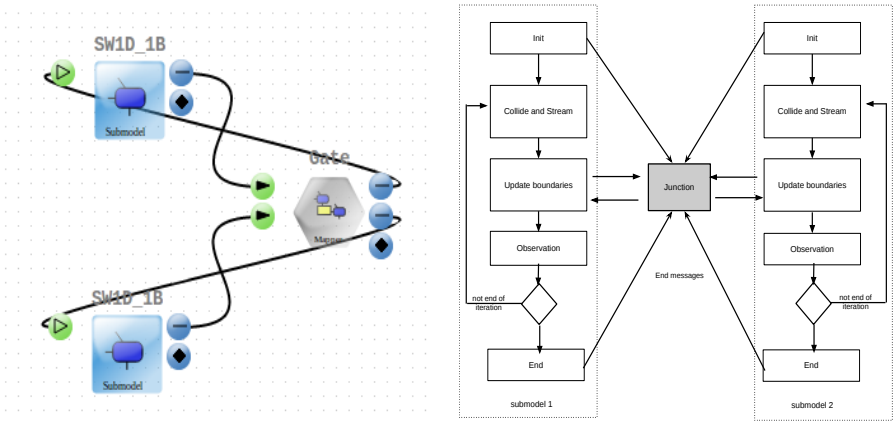
Listing 1.1. Example of CxA file

```
# declare kernels
cxa.add_kernel('SW1D1', 'com.unige.irigcan.kernel.d1.SW1Dkernel')
cxa.add_kernel('SW1D2', 'com.unige.irigcan.kernel.d1.SW1Dkernel')
cxa.add_kernel('Gate', 'com.unige.irigcan.junction.Gate_kernel')
# configure connection scheme
cs = cxa.cs
cs.attach('SW1D1'=>'Gate') {tie('f_out','f1_in')}
cs.attach('SW1D2'=>'Gate') {tie('f_out','f2_in')}
cs.attach('Gate'=>'SW1D1') {tie('f1_out','f_in')}
cs.attach('Gate'=>'SW1D2') {tie('f2_out','f_in')}
#parameters
cxa.env['SW1D1:dx']=0.05
cxa.env['SW1D2:dx']=0.025
cxa.env['SW1D1:dt']=0.025
cxa.env['SW1D2:dt']=0.0125
```

4 Coupling techniques for canal sections

4.1 Tightly coupled sections

A numerical simulation of an entire irrigation canal requires to tightly couple all the submodels with each other. Besides, the multi-scale aspect of our application



(a) The MAD [1] tool, a software part of GridSpace platform, generates automatically the XMML coupling description.

(b) Template of the coupling algorithm for two submodels and a junction.

Fig. 2. A coupling example of two submodels with a gate.

requires a predefined steps of exchanging boundary data between the submodels during simulation. The coupling algorithm for two submodels connected with a junction is described as follow. Each submodel obeys the CxA based formalism proposed in [6], as depicted in Fig. 2(b). In this formalism, a submodel

1. Initializes its parameters from the CxA coupling file, send its maximum number of iterations to the junction, and then starts execution.
2. Performs one step of computation (e.g collide and stream for a LB model)
3. Updates boundary conditions. This is done by sending data to the junction and waiting for updated information.
4. Makes an observation if needed.
5. Increments the iteration counter.
6. Repeats back to step 2 until the maximum number of iterations is reached.
7. Sends *End message* to the junction and finish the execution.

In the present case, the junction is implemented as a daemon program that keeps listening to data from the two corresponding canal sections. It is worth noting that a junction could also be modeled as a full-fledged submodel, with its generic CxA execution loop, solving the boundary condition problem at each iteration and running until the whole execution ends.

The communication process requires synchronization in the following way: the junction uses the *receive()* method, a blocking point-to-point operation of the MUSCLE framework, to receive data from each canal section. From the other side, canals send data to the junction and call the *receive()* method to wait for updated boundary information. Once data has arrived from both canal sections, the junction performs the boundary computation and sends back the updated information to the recipient submodels. This process keeps running until the

maximum iteration number is reached for each submodel. The submodel will then end their execution after sending an *End* message to the junction.

In the case where the upstream and downstream canals have a different spatial and temporal resolutions, grid refinement techniques must be used for the coupling. In addition, the junction must be programmed to handle two different frequencies of send/receive operations for each side. We have implemented such a coupling using the grid refinement algorithm presented in [16]. The temporal resolution Δt_c of the coarse grid was twice greater than the Δt_f of the fine grid. The junction component was running on a separate computer node, thus illustrating the distributed nature of the simulation.

4.2 Coupling through a gate

The information that each submodel (canal section) must send to the junction and that it will receive in return depends on the nature of this junction. Here we show which data and which computation are needed to couple two 1D canal sections that are interconnected through a gate (the junction). This coupling is based on the standard relation [12] giving the water flow Q through a gate, as a function of the water height of the upstream and downstream canals.

$$Q = F(h_A - h_B) = \gamma\sqrt{h_A - h_B} \tag{2}$$

where h_A and h_B are the water heights before and after the gate, respectively and γ is a coefficient depending on the gate opening.

In the LB approach to the SW equation (see section 2), one has (assuming $\Delta x/\Delta t = 1$)

$$\begin{array}{ll} h_A = f_0(A) + f_1(A) + f_2(A) & Q_A = Q = f_1(A) - f_2(A) \\ h_B = f_0(B) + f_1(B) + f_2(B) & Q_B = Q = f_1(B) - f_2(B) \end{array} \tag{3}$$

where $f_i(x)$ denotes the height distribution functions at the point x, and A and B are the points just before and just after the gate, subject to the same discharge Q (see Fig. 3(a)).

Since A and B are at the extremity of the two canal sections, $f_2(A)$ and $f_1(B)$ are unknown. They have to be provided by the gate model which receives $f_0(A)$ and $f_1(A)$ on its left, and $f_0(B)$ and $f_2(B)$ on its right.

After eliminating $f_1(A)$ and $f_2(B)$ from eqs (3), one computes $f_2(A)$ and $f_1(B)$ as

$$f_2(A) = \frac{1}{2}\left[h_A - f_0(A) - Q\right] \qquad f_1(B) = \frac{1}{2}\left[h_B - f_0(B) + Q\right] \tag{4}$$

By subtracting these two equations, one gets

$$f_1(B) - f_2(A) = Q + \frac{1}{2}\left[h_B - h_A + f_0(A) - f_0(B)\right] \tag{5}$$

If we assume that h_A and h_B are known from the previous time step (which is certainly true in a steady state), then the right-hand side of eq. (5) is known

because Q is a known function of $h_A - h_B$. This gives a first equation for $f_1(B)$ and $f_2(A)$.

A second equation is obtained by requesting that the total amount of water that enters the two canal sections is equal to the amount of water that leaves them. This implies that

$$f_1^{in}(B) + f_2^{in}(A) \equiv f_1(B) + f_2(A) = f_1^{out}(A) + f_2^{out}(B) \tag{6}$$

where $f_1^{out}(A)$ and $f_2^{out}(B)$ are known from the last LB step.

Eq. (6) and eq. (5) can be solved for $f_1(B)$ and $f_2(A)$

$$\begin{cases} f_1(B) = \frac{1}{2}\left(\Delta + \bar{Q}\right) \\ f_2(A) = \frac{1}{2}\left(\Delta - \bar{Q}\right) \end{cases} \tag{7}$$

where $\Delta = f_1^{out}(A) + f_2^{out}(B)$ and $\bar{Q} = F(h_A - h_B) + \frac{1}{2}\left[h_B - h_A + f_0(A) - f_0(B)\right]$.

Thus, the *junction component* of our model receives $f_1^{out}(A)$, h_A and $f_0(A)$ from the upstream canal, and $f_2^{out}(B)$, h_B and $f_0(B)$ from the downstream one. It then computes $f_2(A)$ and $f_1(B)$ according to eq. (7) and returns each of these values to each of the canals.

The physical validity of the above coupling can be demonstrated numerically as follows. We enforce the value h_A and h_B in A and B (here we chose $h_A = 0.12$ and $h_B = 0.1$) by imposing at each time step the values of $f_1(A)$ and $f_2(B)$ as

$$\begin{aligned} f_1(A, t+1) &= \alpha f_1(A, t) + (1-\alpha)(h_A(t) - f_2(A, t) - f_0(A, t)) \\ f_2(B, t+1) &= \alpha f_2(B, t) + (1-\alpha)(h_B(t) - f_1(B, t) - f_0(B, t)) \end{aligned} \tag{8}$$

where α is a relaxation parameter (here chosen as $\alpha = 0.2$) used to smooth the way $f_1(A)$ or $f_2(B)$ reach a value that guarantees the prescribed heights h_A and h_B. Note that eq. (8) can be seen as a way to impose a boundary condition for the water height at a chosen canal location.

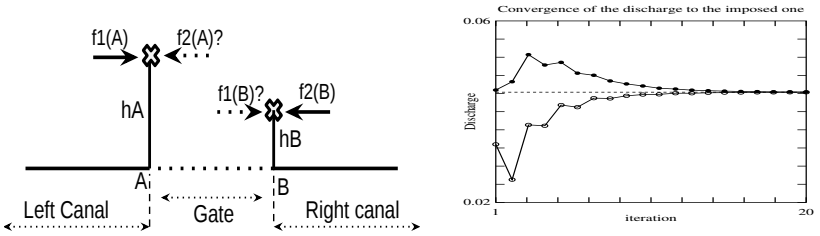
We also specify the discharge Q we want to impose through the gate as $Q = 0.1\sqrt{g(h_A - h_B)} = 0.044294$.

In Fig. 3(b), we show the time evolution of the discharge from arbitrary initial values for $f_2(A)$, $f_0(A)$, $f_1(B)$ and $f_0(B)$. We observe that the quantities $Q_A = f_1(A) - f_2(A)$ and $Q_B = f_1(B) - f_2(B)$ converge rapidly to the imposed value Q. After 20 time steps, Q_A and Q_B are equal to Q within a precision of 10^{-5}. After 40 iterations (not shown), the precision improves to 10^{-8}. After convergence, the value of $\bar{Q}$ is found to be $\bar{Q} = f_1(A) - f_2(B) = 0.035138$, showing that $\bar{Q}$ is clearly different from the imposed discharge Q.

Also, after 40 steps, the value $f_0(A) + f_1(A) + f_2(A) - h_A$ is smaller than 10^{-8}, showing the excellent convergence of the present gate model and the present boundary condition used to impose the heights h_A and h_B. Note that taking $\alpha = 0$ in (8) gives a much slower convergence (oscillations).

5 Conclusion

We have presented a formalism to build an irrigation canal from generic and reusable components, and to simulate the water height and flow over a distributed grid infrastructure. This is obtained by decomposing the corresponding



(a) The known ($f_1(A)$, $f_0(A)$,$f_0(B)$ and $f_2(B)$) and unknown ($f_2(A)$ and $f_1(B)$) distribution functions at the connection lattice site A and B.

(b) Convergence of the flow Q_A (black dots) and Q_B (white dots) to the imposed discharge $Q = 0.044294$. Other parameters: $\tau = 0.6$, $\alpha = .2$, $h_A = 0.12$, $h_B = 0.1$.

Fig. 3. A coupling example of two *SWD1* based submodels with a gate.

canal network into different segments and coupling them through junctions. Each canal segment can be resolved separately with either a 1D shallow water model or 3D, free-surface flow model. The coupling of these segments through junctions have been described in a "standard" way using the MML and CxA formalisms. We have also indicated that our approach can handle relatively different scales, particularly the coupling of coarse and fine LB submodels.

We have proposed an explicit implementation of a gate junction within the LB numerical method. This junction receives information from both canal sections and returns the proper boundary conditions that ensure the right discharge between the two canals.

This demonstrates that our approach of building a canal irrigation based on the MML and CxA formalism can be easily adopted to model much more complex systems. Furthermore, computing results can be sped up by running large simulation as *distributed multiscale application* over an HPC grid infrastructure.

We are currently developing the coupling of 1D and 3D models based on the LB method. Furthermore, we are working on an efficient multi-criteria algorithm to schedule jobs in an intelligent manner that will (1) assign the adequate computing resource to each canal segment accordingly to its computation requirements and (2) reduce communication time and data transfer between coupled segments.

6 Acknowledgment

This research is funded by the European project FP-7 MAPPER under grant agreement RI-261507. Numerical simulations were carried out using the European grid infrastructure. We would like to thank Daniel Harezlak, Eryk Ciepiela and Katarzyna Rycerz for their work and helpful discussions.

References

1. MAD software. `https://gs2.mapper-project.eu/mad/`.
2. MAPPER project. `http://www.mapper-project.eu/`.
3. MUSCLE. `http://developer.berlios.de/projects/muscle/`.
4. OpenPBS: Open Portable Batch System. `http://www.pbsworks.com/`.
5. Palabos toolkit. `http://www.palabos.org/`.
6. A. Hoekstra and E. Lorenz and J.-L. Falcone and B. Chopard. Towards a Complex automata formalism for multuscale modeling. *Int. J. Multiscale Multiscale Computational Engineering*, 5(6):491–502, 2008.
7. Bartosz Bosak and Jacek Komasa and Piotr Kopta and Krzysztof Kurowski and Mariusz Mamoński and Tomasz Piontek. New Capabilities in QosCosGrid Middleware for Advanced Job Management, Advance Reservation and Co-allocation of Computing Resources – Quantum Chemistry Application Use Case. In Marian Bubak, Tomasz Szepieniec, and Kazimierz Wiatr, editors, *Building a National Distributed e-Infrastructure–PL-Grid*, volume 7136 of *Lecture Notes in Computer Science*, pages 40–55. Springer Berlin / Heidelberg, 2012.
8. J. Borgdorff, E. Lorenz, A.G. Hoekstra, J. Falcone, and B. Chopard. A Principled Approach to Distributed Multiscale Computing, from Formalization to Execution. In *e-Science Workshops (eScienceW), 2011 IEEE Seventh International Conference on*, pages 97–104, dec. 2011.
9. B. Chopard and M. Droz. *Cellular Automata Modeling Of Physical Systems*. Aléa-Saclay. Cambridge University Press, 1998.
10. DICE. GridSpace2. `http://dice.cyfronet.pl/products/gridspace`, 2011.
11. Jean-Luc Falcone, Bastien Chopard, and Alfons Hoekstra. MML: towards a Multiscale Modeling Language. *Procedia Computer Science*, 1(1):819–826, 2010.
12. W.H. Graf and M.S. Altinakar. *Hydraulique fluviale: écoulement et phénomènes de transport dans les canaux à géométrie simple*. Traité de génie civil de l'Ecole polytechnique fédérale de Lausanne. Presses Polytechniques Universitaires Romandes, 2008.
13. Jan Hegewald, Manfred Krafczyk, Jonas Tlke, Alfons Hoekstra, and Bastien Chopard. An Agent-Based Coupling Platform for Complex Automata. In Marian Bubak, Geert van Albada, Jack Dongarra, and Peter Sloot, editors, *Computational Science - ICCS 2008*, volume 5102 of *Lecture Notes in Computer Science*, pages 227–233. Springer Berlin / Heidelberg, 2008.
14. A Hoekstra, J L Falcone, Alfonso Caiazzo, and Bastien Chopard. Multi-scale modeling with cellular automata: the complex automata approach. *Cellular automata*, 5191:192–199, 2010.
15. A. G. Hoekstra, A. Caiazzo, E. Lorenz, J.-L. Falcone, and B. Chopard. Complex automata: multi-scale modeling with coupled cellular automata. In *In A. Hoekstra, J. Kroc, and P. Sloot, editors, Modeling complex systems with cellular automata*, volume 3. Springer Verlag, 2010.
16. D. Lagrava, O. Malaspinas, J. Latt, and B. Chopard. Advances in multi-domain lattice Boltzmann grid refinement. *Journal of Computational Physics*, 231(14):4808 – 4822, 2012.
17. Pham van Thang and Bastien Chopard and Laurent Lefèvre and Diemer Anda Ondo and Eduardo Mendes. Study of the 1D lattice Boltzmann shallow water equation and its coupling to build a canal network. *Journal of Computational Physics*, 229(19):7373–7400, 2010.
18. Michael C. Sukop and Daniel T. Thorne. *Lattice Boltzmann Modeling: An Introduction for Geoscientists and Engineers*. Springer, 1 edition, January 2007.