DIFFERENT GOALS IN MULTISCALE SIMULATIONS AND HOW TO REACH THEM

Pierrick Tranouez and Antoine Dutot
LITIS
Université du Havre
UFR Sciences et Techniques
25 rue Ph. Lebon - BP 540
76058 Le Havre Cedex – France
Pierrick.Tranouez@univ-lehavre.fr

AKNOWLEDGEMENTS

Now they're grown-ups and write their articles alone, the authors would like to thank Cyrille Bertelle and Damien Olivier, former PhD supervisors, who could be credited as co-author for all they did on this subject. But they won't. Hey, we must start at some time.

KEYWORDS

Multiscale, clustering, dynamic graphs, adaptation

ABSTRACT

In this paper we sum up our works on multiscale programs, mainly simulations. We first start with describing what multiscaling is about, how it helps perceiving signal from a background noise in a flow of data for example, for a direct perception by a user or for a further use by another program. We then give three examples of multiscale techniques we used in the past, maintaining a summary, using an environmental marker introducing an history in the data and finally using a knowledge on the behavior of the different scales to really handle them at the same time.

INTRODUCTION: WHAT THIS PAPER IS ABOUT, AND WHAT IT'S NOT

Although we delved into different applications and application domains, the computer science research goals of our team has remained centered on the same subject for years. It can be expressed in different ways that we feel are, if not exactly equivalent, at least closely connected. It can be defined as managing multiple scales in a simulation. It also consists in handling emergent structures in a simulation. It can often also be seen as dynamic heuristic clustering of dynamic data[1]. This paper is about this theme, about why we think it is of interest and what we've done so far in this direction. It is therefore akin to a state of the art kind of article, except more centered on what we did. We will allude to what others have done, but the focus of the article is presenting our techniques and what we're trying to do, like most articles do, and not present an objective description of the whole field, as the different applications examples could make think : we're sticking to the same computer science principles overall. We're taking one step back from our works to contemplate them all, and not the three steps which would be necessary to encompass the whole domain, as it would take us beyond the scope of the conference.

[1] We will of course later on describe in more details what we mean by all this.

PERCEPTION: FILTERING TO MAKE DECISIONS

I look at a fluid flow simulation but all I'm interested in is where does the turbulence happen, in a case where I couldn't know before the simulation (Tranouez et al. 2005). I use a multi-participant communication system in a crisis management piece of software and I would like to know what are the main interests of each communicant based on what they are saying (Lesage et al. 1999). I use an IBM model of different fish species but I'm interested in the evolution of the populations, not the individual fish (Prevost et al. 2004). I use a traffic simulation with thousands of cars and a detailed town but what I want to know is where the traffic jams are (coming soon).

In all those examples, I use a piece of software which produces huge amounts of data but I'm interested in phenomena of a different scale than the raw basic components. What we aim at is helping the user of the program to reach what he is interested in, be this user a human (Clarification of the representation) or another program (Automatic decision making). Although we're trying to stay general in this part, we focused on our past experience of what we actually managed to do, as described in "Some techniques to make these observations in a time scale comparable to the observed", this isn't gratuitous philosophy.

Clarification of the representation

This first step of our work intends to extract the patterns on the carpet from its threads (Tranouez 1984). Furthermore, we want it to be done in "real (program) time", meaning not a posteriori once the program is ended by examining its traces (Servat et al; 1998), and sticking as close as possible to the under layer, the one pumping out dynamic basic data. We don't want the discovery of our structures to be of a greater time scale than a step of the program it works upon.

How to detect these structures? For each problem the structure must be analyzed, to understand what makes it stand out for the observer. This implies knowing the observer purpose, so as to characterize the structure. The

answers are problem specific, nevertheless rules seem to appear.

In many situations, the structures are groups of more basic entities, which then leads to try to fathom what makes it a group, what is its inside, its outside, its frontier, and what makes them so.

Quite often in the situation we dealt with, the groups members share some common characteristics. The problem in that case belongs to a subgenre of clustering, where the data changes all the time and the clusters *evolve* with them, they are not computed from scratch at each change.

The other structures we managed to isolate are groups of strongly communicating entities in object-oriented programs like multiagent simulations. We then endeavored to manage these cliques.

In both cases, the detected structures are emphasized in the graphical representation of the program. This clarification lets the user of the simulation understand what happens in its midst. Because modeling, and therefore understanding, is clarifying and simplifying in a chosen direction a multi-sided problem or phenomenon, our change of representation participates to the understanding of the operator. It is therefore also a necessary part of automating the whole understanding, aiming for instance at computing an artificial decision making.

Automatic decision making

Just like the human user makes something of the emerging phenomena the course of the program made evident, other programs can use the detected organizations.

For example in the crisis management communication program, the detected favorite subject of interest of each of the communicant will be used as a filter for future incoming communications, favoring the ones on connected subjects. Other examples are developed below, but the point is once the structures are detected and clearly identified, the program can use models it may have of them to compute its future trajectory. It must be emphasized that at this point the structures can themselves combine into groups and structures of yet another scale, recursively. We're touching there an important component of complex system. We may hope the applications of this principle to be numerous, such as robotics, where perceiving structures in vast amounts of data relatively to a goal, then being able to act upon these accordingly is a necessity.

We're now going to develop these notions in examples coming from our past works.

SOME TECHNIQUES TO MAKE THESE OBSERVATIONS IN A TIME SCALE COMPARABLE TO THE OBSERVED

The examples of handling dynamic organization we chose are taken from two main applications, one of a simulation of a fluid flow, the other of the simulation of a huge cluster of computing resources, such as computers. The methods titled "Maintaining a summary of a simulation" and "Reification: behavioral methods" are theories from the hydromechanics simulation, while "Ants" refers to the computing resources management simulation. We will first describe these two applications, so that an eventual misunderstanding of what they are doesn't hinder later the clarity of our real purpose, the analysis of multiscale handling methods.

In a part of a more general estuarine ecosystem simulation, we developed a simulation of a fluid flow. This flow uses a particle model (Leonard 1980), and is described in details in (Tranouez 2005) or (Tranouez et al. 2005). The basic idea is that each particle is a vorticity carrier, each interacting with all the others following Biot-Savart laws. As fluid flows tend to organize themselves in vortices, from all spatial scales from a tens of angstrom to the Atlantic Ocean, this is these vortices we tried to handle as the multiscale characteristic of our simulation. The two methods we used are described below.

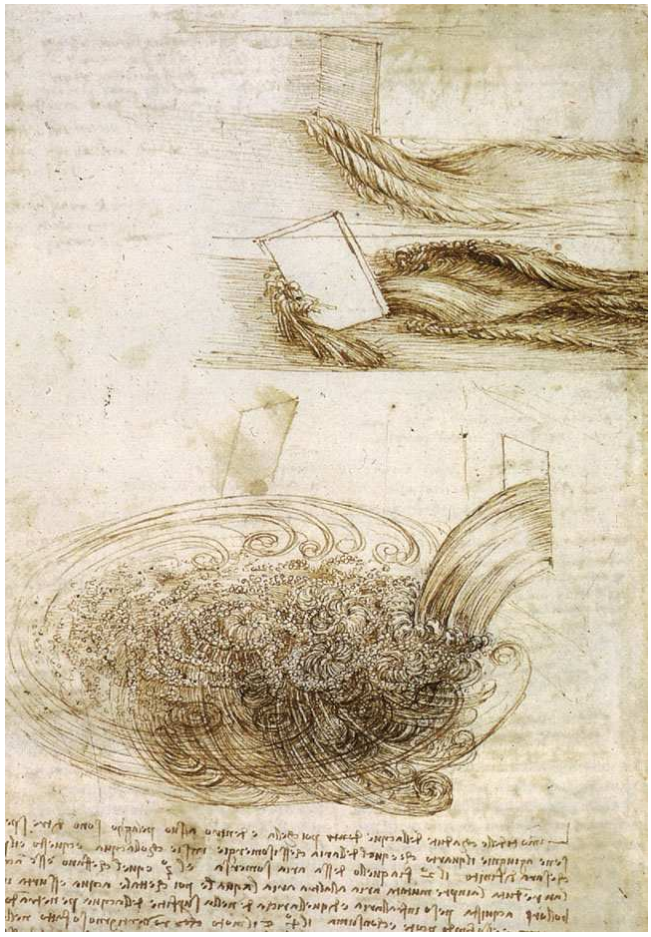

Figure 1: Vortices in a fluid flow by Leonardo Da Vinci

The other application, described in depth (Dutot 2005), is a step toward automatic distribution of computing over computing resources in difficult conditions, as:

- The resources we want to use can each appear and disappear, increase or decrease in number.

- The computing distributed is composed of different object-oriented entities, each probably a thread or a process, like in a multiagent system for example (the system was originally imagined for the ecosystem simulation alluded to above, and the entities would have been fish, plants, fluid vortices etc., each acting, moving …)

Furthermore, we want the distribution to follow two guidelines:

- As much of the resources as possible must be used,
- Communications between the resources must be kept as low as possible, as it should be wished for example if the resources are computers and the communications therefore happen over a network, bandwidth limited if compared to the internal of a computer.

This the ultimate goal of this application, but the step we're interested in today consists in a simulation of our communicating processes, and of a program which, at the same time the simulated entities act and communicate, advises how they should be regrouped and to which computing resource they should be allocated, so as to satisfy the two guidelines above.

Maintaining a summary of a simulation

The first method we would like to describe here relates to the fluid flow simulation. The hydrodynamic model we use is based on an important number of interacting particles. Each of these influences all the other, which makes n^2 interactions, where n is the number of particles used. This makes a great number of computations. Luckily, the intensity of the influence is inversely proportional to the square of the distance separating two particles. We therefore use an approximation called fast multipoles method, which consists in covering the simulation space with grids, of a density proportional to the density of particles (see Figure 2). The computation of the influence of its colleagues over a given particle is then done exactly for the ones close enough, and averaged on the grid for those further. All this is absolutely monoscale.

As the particles are vorticity carriers, it means that the more numerous they are in a region of space, the more agitated the fluid they represent is. We would therefore be interested in the structures built of close, dense particles, surrounded by sparser ones. A side effect of the grids of the FMP, is that they help us do just that. It's not that this clustering is much easier on the grids, it's above all that they are an order of magnitude less numerous, and organized in a tree, which makes the group detection much faster than if the algorithm was ran on the particles themselves. Furthermore, the step by step management of the grids is not only cheap (it changes the constant of the complexity of the particles movement method but not the order) but also needed for the FPM.

We therefore detect structures on

- Dynamic data (the particles characteristics)
- With little computing added to the simulation,

which is what we aimed at.

The principle here is that through the grids we maintain a summary of the simulation, upon which we can then run static data algorithm, all this at a cheap computing price.

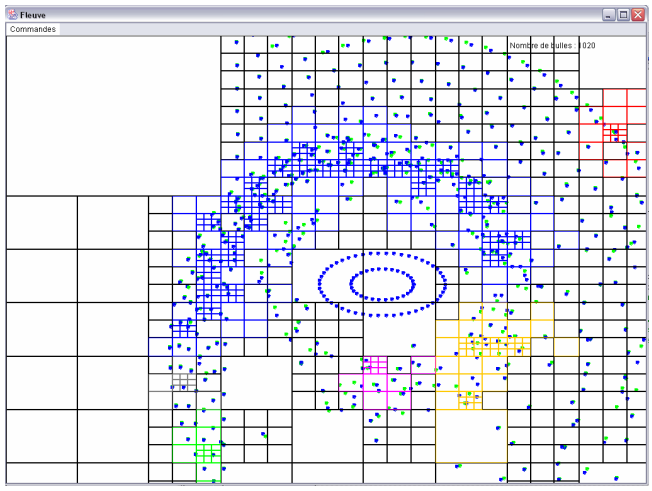

Figure 2 : Each color corresponds to a detected aggregate

Ants

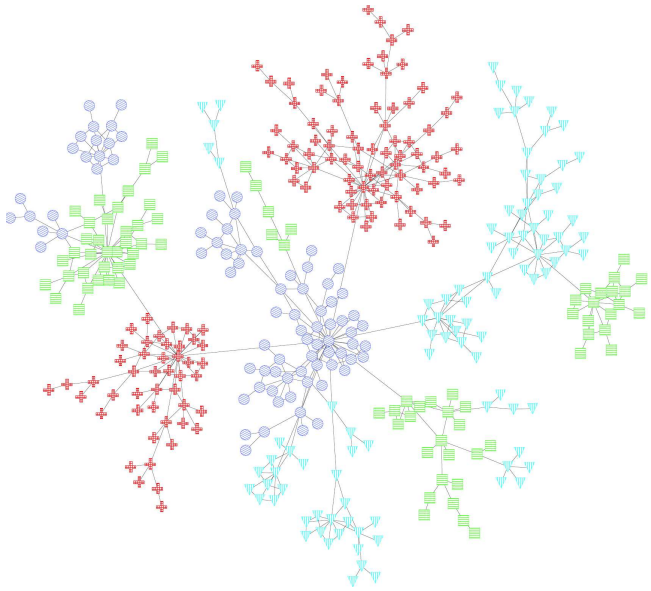

Figure 3: Each color corresponds to a detected aggregate

Reification: behavioral methods

This last example of our multiscale handling methods was also developed on the fluid flow simulation. Once more, we want to detect structures in a dynamic flow of data, without getting rid of the dynamicity by doing a full computation on each step of the simulation. The idea here is doing the full computation only once in a good while, and only relatively to the unknown parts of our simulation.

We begin with detecting vortices on the basic particles once. Vortices will be a rather elliptic set of close particles of the same rotation sense. We then introduce a multiagent system of the vortices. We have indeed a general knowledge of the way vortices behave. We know they move like a big particle in our Biot-Savard model, and we model its structural stability through social interactions with the surrounding basic particles, the other vortices and the obstacles, through which they can grow, shrink or die (be dissipated into particles). The details on this can be found in (Tranouez et al. 2005). Later on we occasionally make a full-blown vortex detection, but only on the remaining basic particles, as the already detected vortexes are managed by the multiagent system

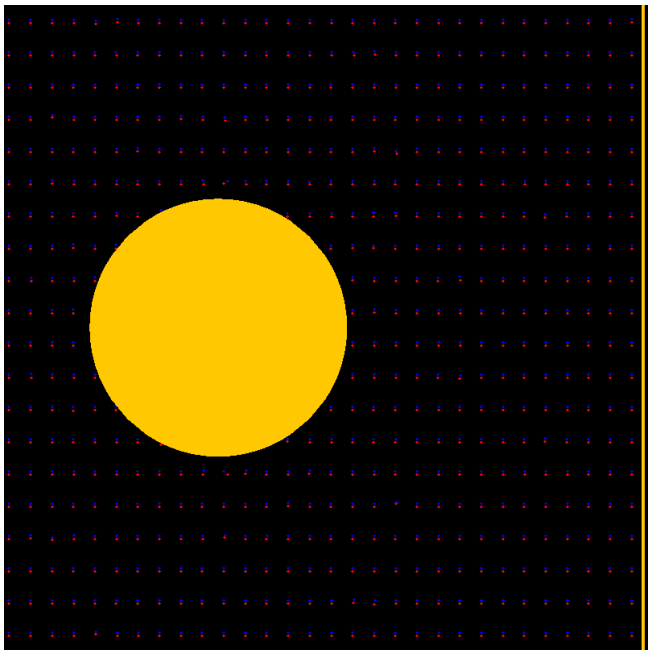

Figure 4: Fluid flow around an obstacle, initial state

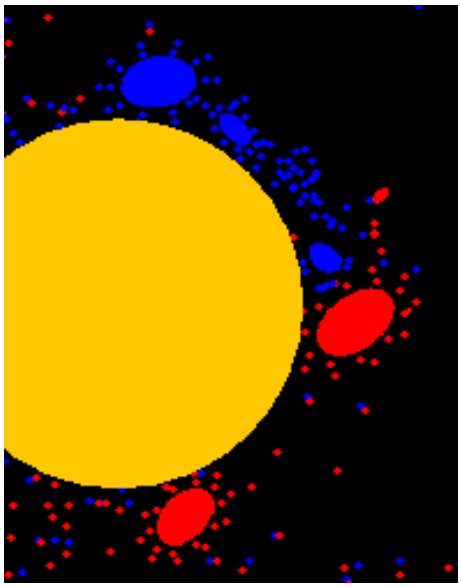

Figure 5: Part of the flow, some steps later. The ellipses are vortices.

In this case, we possess knowledge on the structures we want to detect, and we use it to actually build the upper scale level of the simulation, which at the same time lightens ulterior structures detection. We're definitely in the category described in Automatic decision making.

CONCLUSION

Our research group works on complex systems and focused on the computer representation of their hierarchical/holarchical characteristics (Koestler 1978) (Simon 1996) (Kay 2000). We tried to illustrate that describing a problem at different scales is a well-spread practice at least in the modeling and simulating community. We then presented some methods for handling the different scales, with maintaining a summary, using an environmental marker introducing an history in the data and finally using a knowledge on the behavior of the different scales to really handle them at the same time.

We know believe we start to have sound multiscale methods, and must focus on the realism of the applications, to compare the sacrifice in details we make when we model the upper levels rather than just heavily computing the lower ones. We save time and lose precision, but what is this trade-off worth *precisely*?

REFERENCES

Dutot A., *Distribution dynamique adaptative à l'aide de mécanismes d'intelligence collective*, PhD thesis, Le Havre University

Kay J., « Ecosystems as Self-Organising Holarchic Open Systems : narratives and the second law of thermodynamics », S.E.JORGENSEN, MÜLLER F., Eds., *Handbook of Ecosystems Theories and Management*, Lewis Publishers, 2000.

Koestler A. 1978, *Janus. A Summing Up* 1978

Leonard A. 1980, « Vortex methods for flow simulation », *Journal of Computational Physics*, vol. 37, 1980, p. 289-335.

Lesage F., Cardon A., Tranouez P. : "A multiagent based prediction of the evolution of knowledge with multiple points of view"; KAW' 99; (1999)

Prevost G., Tranouez P., Lerebourg S., Bertelle C., Olivier D. 2004 : "Methodology for holarchic ecosystem model based on ontological tool"; ESMC 2004; pp 164-171 (2004)

Servat D., Perrier E., Treuil J.-P., Drogoul A. 1998, « When Agents Emerge from Agents: Introducing Multi-scale Viewpoints in Multi-agent Simulations », *MABS*, 1998, p. 183-198.

Simon H. 1996, *The Sciences of the Artificial (3rd Edition)* MIT Press

Tranouez Pierre 1984, *Fascination et narration dans l'œuvre romanesque de Barbey d'Aurevilly* , Doctorat d'État

Tranouez Pierrick 2005, *Penicillo haere, nam scalas aufero*, PhD thesis, Le Havre University

Tranouez P., Bertelle C. and Olivier D. 2005, « Changing levels of description in a fluid flow simulation", EPNADS 2005