OO Reactive Agents for RDM-Based Simulations

S.Bandini, F. De Paoli, S. Manzoni
DISCO - Università di Milano Bicocca
Via Bicocca degli Arcimboli 8, 20126, Milano
{bandini, depaoli, manzoni}@disco.unimib.it

C. Simone
DI - Università di Torino
Corso Svizzera 185, 10149, Torino
simone@di.unito.it

Abstract

The computer simulation of complex phenomena is a challenging issue for studying their properties. Several models and techniques have been developed in order to provide useful conceptual and computational frameworks.

The aim of this paper is to present a simulation model based on the Reaction-Diffusion Machine (RDM) and to discuss its implementation by a system based on multi-agent and Object Oriented paradigms. The chemical extraction of substances in washing phenomena occurring in percolation processes has been considered as reference application.

1. Introduction

Many fundamental structures and dynamical behaviours in physics, chemistry and biology are described in terms of reaction-diffusion models or metaphors that are rooted in the work of Alan Turing [7]. Applications of this model range from pattern formation or epidemic spreads to natural selection through ecological systems and the percolation systems. Reaction refers to phenomena where two or more entities (agents) become in contact and modify their state in consequence of this fact. Diffusion implies the existence of a space where the involved agents are situated and can move. While reaction is univocally interpreted, diffusion takes different meanings in different disciplines. Specifically, in the disciplines which consider biotic agents, diffusion is interpreted as movement of agents in a given space: for example, in animal population dynamics [5], agents move to concentrate together or to scatter according to behavioural patterns and mutual interactions. As such, reaction and diffusion describe the behaviour of agents in terms of changes of state and position, respectively; however, they are not enough to describe that some factor, possibly independent of the involved agents, can influence the way in which they perform both reaction and diffusion behaviours. This influence can be captured by the notion of *field* as used in physics. There are fields whose sources are outside the considered space: for example, the heat, light, gravity; and fields that are originated by some agents located in the space: for example, the emission of ad hoc substances in the case of animal species (like pheromone in ants) or the heat and light generated by a lamp. Both kinds of fields are characterised by specific distribution laws that determine fields' values at each point in the space; moreover, agents are characterised by their capability of being sensitive to those fields at different degrees in relation to their current state.

The reaction diffusion machine model allows for the simulation of complex systems in which entities react locally with each other and with the environment, and the global system behaviour emerges from the local behaviour of the composing entities. In the reaction-diffusion machine (RDM) [1] the control is fully distributed. The agent behaviour is determined by a local 'computation' based on its position and sensitivity to fields as well as on reaction and diffusion patterns characterising its type.

The RDM simulation model has already been implemented on a parallel computer [2]. The objective of that implementation was to deliver an effective tool to be used to collect information on realistic simulations (i.e. to explicitly represent topological features of the simulated phenomena), which means that it was designed to process thousands, or even millions, of data to represent a significant portion of the real experiment. The model proved to be accurate and effective, and the tool demonstrates to scale over the simulation size. The simulation time resulted in direct inverse relation to the number of processors.

This paper describes the development of an agent-based implementation that conforms more accurately to the distributed nature of the model and, in addition, provides a portable simulation tool with minimum loss of performance.

The object-oriented paradigms is suitable for agent implementations since it models the world by independent entities, the *objects*, that are autonomous and communicate with each other by means of messages. Moreover, available object-oriented platforms allow objects to be autonomous, and to be distributed over a computer network. These properties facilitate the task of developing agent systems [8]. Literature reports successful

stories about agent-based systems for scientific computing [4]. Anyway, the use of object-oriented platforms for scientific simulation is still questionable, since the costs of the infrastructure to support the execution of an object-oriented system could be too costly in term of performance. One of our goals is to evaluate this aspect by benchmarking the parallel implementation and the distributed, object-oriented implementation of the simulator.

As reference application, we consider the chemical extraction of substances in washing phenomena occurring in percolation processes. This kind of phenomena are usually modelled and simulated by numerical approaches. Moreover, cellular automata approaches have been developed [3]. A comparison between RDM and cellular automata has been discussed in [1].

The next section presents the reaction diffusion machine. Section 3 describes the chemical problem that we adopted as reference case study for our developments. Section 4 discusses the concepts behind the agent system we are designing to simulate the chemical phenomenon. Section 5 outlines the object-oriented design of the simulation tool. Finally, section 6 concludes the papers with a description of the ongoing activities.

2. The reaction diffusion machine model

The Reaction-Diffusion Machine model [1] is defined by a *space*, a set of *sites* with an *adjacency relation* among them, a set of *reactive agents* populating the space, and an *interaction mechanism* among agents.

Their type distinguishes agents. The type of an agent specifies the values of some parameters that govern its behaviour in given *states*. These parameters are: a *field sensitivity function* telling in which states the related entity is sensible to which fields in accordance with a vector of *thresholds*, one value for each field. A *field* is characterised by its set of *values* and *distribution function* and by functions to compose and compare field values, sensitivity values and thresholds. Notice that each agent can be source of, and sensitive to, different fields at the same time. The perceived fields can be generated by agents. An agent is characterised by its type, current state and position. A *general law* states that each site contains at most one agent (non-copenetrability law).

The dynamic part of the model is described in terms of *rule schemes* that define the transitions between configurations (as in standard transition systems). There are four main rule schemes are based on both state and position of the involved agents:

• *Field diffusion rules* define when a field is generated (i.e., if an agent of a specific *type*, in a specific *state*, and is a *source* of a *field*, then the field propagates according to its *distribution function* and *initial value).*

• *Trigger rules* define how a field effects the state of an agent (i.e., if an agent is sensitive to a *field* in relation to its *state*, *threshold* and *sensitivity function*, then the agent changes its state according to a predefined *state transition function).*

• *Transport rules* define how a field effects the position of an agent (i.e., if an agent is sensitive to a specific *field* - as for trigger rules - then it changes its location according to a predefined *site transition function*).

• *Reaction rules* define the synchronous state change of agents constituting a *vicinity*, that is, they are all pairwise adjacent (i.e., if agents constituting a *vicinity* possess specific *types* and specific *states*, then they change their state according to a predefined *state transition function*).

3. The chemical extraction processes

The application problem that has been used to test the RDM model concerns chemical reactions and the movement of fluid particles in a porous medium. The application field is the pesticide percolation through soil (the application has been developed in collaboration with the International Centre for Pesticide Safety).

Within the general framework of the problem of pesticides pollution in the soil, we focused our attention on the problem of pesticide's leaching to ground water caused by water percolation through the soil. When applied to crops, pesticides are absorbed by soil. Then, when water flows through the soil (percolates) because of rain or floods, pesticides can be released into it. Water containing pesticides reached the groundwater layer because of gravity, and, since groundwater is usually the source of common tap water, it is straightforward to understand the pollution danger deriving from the excessive use of pesticides [6]. The amount of pesticide released changes according to the chemical properties of the pesticide itself and the physical and morphological properties of the soil.

The extraction process of soluble substances (pesticides) from the percolation bed (soil) can be divided into two main phases: washing and diffusion. Washing corresponds to the reaction that takes place between water flowing into the percolation bed and the surface of the soil particles. This phenomenon causes the release of pesticide from the soil to the water. Diffusion is characterised by the uniform spreading into the water of the chemicals washed from the percolation bed.

The main goal of the simulation of the percolation process is to obtain the flow rate of the pesticide in different condition (e.g., granulometric distribution of the particles composing the ground). The simulation model based on RDM has been created in a multi-reactive agent

perspective, where the entire extraction process is modelled in terms of a complex collective phenomenon.

4. The RDM agent-based model for the simulation

The space is modelled as a regular two-dimensional grid. Two types of agents are introduced*: S-Agents* (representing soil particles) and *W-Agents* (representing single portions of water). S-Agents are motionless and sited on sites. W-Agents are mobile and can move on free sites on the grid. Moreover, they move reacting to an external field (gravity). S-Agents can interact only with W-Agents. W-Agents can interact either with other W-Agents or with S-Agents (asymmetric interaction behaviour)

Both the types of agents (W and S) possess a tank of particles whose capacity is defined by saturation constant. Each tank is divided into as many compartments as the number of the adjacencies of each agent on its site (four in the case of the regular grid). Each compartment contains a portion of the quantity of particles that can be exchanged only with the adjacent compartment of an adjacent agent.

From the standpoint of the interaction among agents, S-Agents can only give particles to W-Agents (equilibrium law), and W-Agents can absorb or give particles from and to other W-Agents and only absorb particles from S-Agents (same law).

The behaviour sets of rules determining the dynamics of the entire multi-agent system are:

Reaction Rules: Recognition of free/taken sites in the surrounding of radius 1 and interaction among agents (S-Agents/W-Agents, W-Agents/W-Agents) to exchange particles according to an “equilibrium law”.

Balance Rules: The total quantity of the particles of the 4 compartments of each agent's tank is uniformly redistributed.

Movement Rules: W-Agents can move only to south, east or west free sites.

5. The Object-Oriented multi-agent system

The RDM simulation model described in the previous sections has been implemented on a parallel computer to be used as tool to collect realistic information on percolation phenomena. That implementation was written in C according to the MPI model over a Cray computer with up to 128 processors. The major goal of that development was performance maximization, since that is a crucial property of any effective simulation tool. The experiment was successful since the simulation time resulted in direct inverse relation to the number of processors.

The purpose of the multi-agent implementation is to model the reality more accurately and provide a portable simulation tool with little loss of performance. For these reasons, we choose to develop the new prototype directly in the Java programming language and platform, instead of on specialized platforms for agent systems. Java is object oriented, which means that entities can be modeled by *objects* to capture their properties in a single component. Java is multithreaded, which means that each object can have an associated thread. Therefore, it is suitable for developing autonomous components. Java ensures portability, since it runs on a virtual machine that is largely available over the majority of systems. Moreover, Java is *network enabled*, which means that objects and classes can be transferred over a computer network. All these peculiarities make Java a perfect tool to develop agents systems.

The Java simulation tool is composed of objects to model the environment and the two types of agents described above.

Passive objects that implement the grid and control the simulation execution model the environment. A matrix that hosts the agents implements the simplest grid. Each agent is aware of its position on the grid and, therefore, can access its adjacent elements. Communication can occur directly between two agents. The movement is implemented by the transfer of an agent from the current position to a next position in the matrix. A more sophisticated grid layout is implemented by a set of objects, named *sites*. They are connected according to the topology of the percolation bed. The state of every site holds information about the gravity external field, the adjacent sites and the hosted agent, if any. A site can answer messages about its state information.

The former solution should be more efficient, since the infrastructure is kept at the minimum, but it has the drawback of being feasible only in a single-processor simulation. The latter forms a network of hosting objects that can be distributed over a set of processors. Moreover, this solution fits better the physical model since the agent can find locally -by asking the host site- information about the environment. In the former solution, instead, the agents need to hold information on their position and the gravity field. We decided to implement both the solutions to compare them with each other and with the C implementations.

The two type of agents, S-agent and W-agent, are modelled as active objects. The state of an agent represents the tank and its four partitions to define the amount of pesticide it can carry on. The interaction between agents is modelled by message exchanging.

Agents are active, since they have to react, balance and move, as described above. Autonomous threads of execution are associated with active objects to let them act

concurrently. The activity carried on by a W-agent can be summarised as follows:

(a) it checks with the adjacent sites whether they host agents, and, if that is the case, it reacts with the adjacent agents by issuing messages;
(b) it balances its internal state;
(c) it moves on the bases of the information about empty sites next to it, and the local gravity field;

The above activities are repeated indefinitely. S-agents differ from W-agents by the fact that they do not perform activity (c).

The reaction activity is implemented by a set of message exchange to let each of the involved agents know the number of particles of the adjacent partitions. This information allows the agent to define the new number of particles for each tank partition, thus simulating the particles exchange.

The RDM model is synchronous, which means that agents need to synchronise before reacting each other. Synchronisation has been modelled by the definition of an object, named Synchroniser, which grants each agent the permission of proceeding before and after the reaction activity. The Synchroniser is a shared object that ensures full synchronous behaviour on a single processor machine. In a distributed implementation, the full synchronisation constraint has to be released in favour of a local synchronisation mechanism that involves clusters of adjacent agents.

Although it involves a single agent, diffusion activity requires synchronisation, since agents' moving may result in a conflict when two or more agents decide to occupy the same site. To model the effect of the gravity field - which is not explicitly modelled, since it is constant in each site-, we assume that moving to a South-site is always granted, while there is not a preferred way between moving to West-sites or East-sites. Therefore, a first-come-first-served policy can be adopted. This policy can be easily implemented by having an agent lock the involved portion of the grid before performing the diffusion activity. In the single processor implementation this results in a mutual exclusive access to the grid, while in the distributed implementation it results in a lock of the involved site.

6. Conclusions

The RMD model has been proved effective for the class of problems addressed in this paper. This has been experimentally proved through a parallel implementation of a cellular-automata-based model of this complex system [2]. A development based on the multi-agent and object-oriented paradigms should deliver a clean and flexible platform that can be easily tailored to accommodate several kinds of simulations to fit specific domain requirements. The implementation is developed in Java to ensure portability and open the possibility of different execution patterns, ranging from concurrent execution on a single virtual machine to a truly distributed execution over networked computers.

Preliminary versions of both single-processor and multi-processor implementations are available for testing. The current activity is twofold: complete the testing phase of the two versions and work on the multi-processor version to release the constraints introduced in the other versions due to the assumptions of centralised control. This evolution should deliver a platform in which agents move and act freely, without other synchronisation then the ones requested by the reaction with the *vicinity*.

Moreover, benchmarks are planned to contrast the multi-agent implementation in Java of the simulation tool on networked workstations with the implementation in C on a parallel computer. Even if it is likely that the parallel implementation overtakes the multi-agent implementation, it will be interesting to evaluate what is the performance decrease. The conjecture is that the multi-agent system is still effective to run realistic simulation over a number of data. To prove it, we will evaluate whether the gain in design and implementation quality (in terms of software engineering principles) and in simplicity and economy (in terms of computer availability and costs) compensates the loss in performance.

References

[1] S. Bandini, and C. Simone, “Integrating Forms of Interaction in a Distributed Coordination Model”, *Fundamentae Informaticae*, in press.

[2] S. Bandini, G. Mauri, G. Pavesi, C. Simone, “Parallel Simulation of Reaction-Diffusion Phenomena in Percolation Processes: a Model Based on Cellular Automata”, *Future Generation Computer Systems*, in press.

[3] B. Chopard, M. Droz, *Cellular Automata Modelling of Physical Systems*, Cambrige University Press, 1998.

[4] R. Gustavsson, “Networked Agents for Scientific Computing”, *Communication of the ACM*, Vol. 42, N. 3, March 1999.

[5] A.Okubo, *Diffusion and Ecological Systems: Mathematical Models*, New York: Springer Verlag, 1980.

[6] P. S. C. Rao, A. G. Hornsby, and R. E. Jessup, “Indices for Ranking the Potential for Pesticide Contamination of Groundwater”, TR ICPS, 1985.

[7] A. Turing, “The Chemical Basis of Morphogenesis. Philo”s. *Trans. R. Society*, vol. 237, 1952.

[8] A. M. Uhrmacher, “Concepts of Objects – and Agent Oriented Simulation”, *Transactions on SCS*, Vol. 14. No. 2, 59-67, 1997.