An Autonomous Agent System to Simulate a Set of Robots Exploring a Labyrinth

Nourredine BENSAID and Philippe MATHIEU
Laboratoire d'Informatique Fondamentale de Lille
U.R.A. 369 du C.N.R.S, Université de Lille I,
59655 Villeneuve d'Ascq Cedex. FRANCE.
e-mail :{bensaid,mathieu}@lifl.fr
Phone: +33 (0)3 20 43 69 02, Fax: +33 (0)3 20 43 65 66

Abstract

This work presents an application of the agent technology to the simulation in the robotic domain. A group of robots cooperate with each other to simulate a large surface which has a labyrinth structure. A robot is represented as a specialist agent which has for task to recognize different paths characterizing this surface. The application is supported by a hybrid and hierarchical architecture model called MAGIQUE[1]. The interactions between robots are managed by *supervisor* agents which have a global vision of the labyrinth. The explored parts of the labyrinth are hold at a blackboard structure managed by the supervisor. Agents can communicate either directly by message-passing or indirectly via their supervisor. The paper begins by describing the application and presents different constraints related to the move of the robots. It discusses how the application is implemented in a distributed programming environment and presents also the practical constraints faced in the instantiation of the model. The application addresses also an issue related to the load balancing between robots. Finally, it concludes by investigating a set of issues to explore.

Topic Area: Multiagent systems

Introduction

Many works so far have been achieved on multiagent domain, particularly aspects related to the agent theories, architectures and languages. This rich background (Cohen & Levesque 1990) (Wooldridge 1995) (Muller & Pischel 1993) (Shoham 1993) aims to provide a set of theoretical tools intended to build frameworks for designing, specifying and programming intelligent agents and multiagent systems. All the previous tools ensure theoretical foundations for outstanding systems supporting complex applications. Thus, several application areas are concerned with the agent paradigm. Typical applications are information retrieval over the Internet, scheduling meetings, intelligent robotics, integration of heterogeneous software agent and, business and industrial process modeling. As the term "agent" is used in many different ways, we will clarify what we mean by it. An agent is a computational process which has a set of capabilities for reasoning, perceiving and acting on its environment. It is represented by its mental attitudes: beliefs, goals and capabilities. The development of applications based on agent technology becomes an attractive challenge to consider. In this context, our contribution consists in using the model MAGIQUE (Bensaid & Mathieu 1997) to simulate a group of robots exploring different paths of a labyrinth. The idea originates from the metaphor which consists in giving out from an air-engine a set of robots over an unexplored planet. Robots start the exploration from different points of the labyrinth. Each robot has for task to pursue one direction until there is no issue to move. In this situation, the robot has found one path and then informs its acquaintances that this one is explored. If the robot has other alternatives to explore, it repeats the previous process, otherwise it causes a cooperation with its acquaintances in order to find another point from where it can start the exploration. Before starting the exploration, a robot must ensure that there is at least one path issued from this point, which is not explored. Then, a robot must know all paths which are already explored by its acquaintances. It has a limited vision of the environment. It can just perceive whether its neighbour points are obstacles or no. In the case where there are many neighbour points which are not obstacles, the robot chooses one point to explore and registers other alternative points. Once the path completely explored, the robot should come back until an alternative point and reiterates the same process. A supervisor agent has a global vision of the labyrinth. Whenever all agents have finished the exploration of their respective area, the supervisor indicates them another unexplored one in the labyrinth. The cooperation between robots arises when one robot has finished the exploration of its area and there is not another new area to explore. In this case, it communicates with the more loaded robot. A robot is considered as more loaded if it has the greatest alternative points to explore. The cooperation will success if there is a path between areas explored by both the loaded robot and the least loaded one. MAGIQUE is used as a framework to implement this applica-

[1]MAGIQUE means a Hybrid and Hierarchical Multi-Agent Architecture Model

tion because all system's robots share a same labyrinth and solutions obtained (paths explored) are hold at the blackboard. The latter is managed by a supervisor of a group of robots. Thus the blackboard contains the current solution of the problem, and all robots should consult the blackboard to progress in their exploration. On the other hand, a robot should interact directly with its acquaintances in order to get knowledge, particularly when robots cooperate to make the load well balanced. Another application can be supported by MAGIQUE is the cooperation between multiple inference systems written in different languages by different peoples. Each supervisor of a group comprises tools allowing to heterogeneous agents to communicate with each other. Thus, It contains a dictionary used to translate requests submitted by an agent to another one.
The paper is organized as follow:
In the next section, the application is described in detail. The entities of the system are presented and the structure of the agent is given. In the following section, the content of communicative actions between agents is specified. The load balancing issue is addressed through this section because the supervisor has a global vision of the load of robots and then identify instantaneously the more loaded robot. The last section comprises details of the application implementation. It describes the programming environment and constraints related to the communication between agents, as the implementation of a blackboard in a distributed environment. We assume some familiarity with concepts closely tied to distributed systems as *thread*.

Description of the application

The robots are represented by specialist agents which have a set of properties. A specialist is capable of perceiving its environment and thus, knowing whether a position at its neighbourhoud is an obstacle or not. If there are many points at its neighbourhoud which are not obstacles, the specialist moves in one random-chosen direction and registers all other alternative points in its belief base. Each position of the labyrinth which is not an obstacle has two states: *explored* or *unexplored*. During its progress in the exploration process, the specialist does not take into account a position which either is explored or is an obstacle. Since specialists access to the labyrinth and modify the state of each position, the labyrinth is considered as a second blackboard of the system. All the alternative points made up the robot environment model. The belief-rule base consists in backtracking the exploration process and then removing from the belief base, the point from where the exploration was started. The cooperation base allows the supervisor to know the load of a specialist if the number of alternative points grows considerably. The exploration process consists in performing a capability called *explore(P)* which enables the robot to begin the exploration from the point *P*. When a robot is loaded (contains many alternative points), it requests its supervisor which must find him another robot, either available or least loaded, with which it will cooperate. In the second case (i.e. the acquaintance is a least loaded robot), the more loaded robot sends to the least one a list of alternative points to explore. The cooperation between the two robots is possible if there is one alternative point, in the submitted list, from where there is an explored path to the explored area of the least loaded robot. Therefore, the least loaded robot must consult the blackboard to determine if there is a path between its exploration zone and the one of the more loaded one. This process is achieved by performing the robot capability *find_path*. The supervisor coordinates interactions of robots. Its belief base contains the model of the group of robots. The information related to the load of robots, their state (available or active) are hold at the supervisor. Furthermore, the supervisor manages the blackboard containing the current solution. The blackboard is a structure comprising all explored paths. Whenever a robot has finished the exploration of its area, it requests its supervisor which updates its belief base. If there is another area which is not explored, the supervisor will activate an available robot to start the exploration, otherwise it finds the more loaded robot and the least loaded one, then releases the cooperation process between them.
In addition to both the blackboard containing the current solution and the blackboard representing the labyrinth, the supervisor contains two data structures where it holds the load of robots and their state (agent available or active). It has also capabilities to serialize conflicting accesses to the blackboard. Agents use communicative actions, *request* and *inform*, similar to those proposed in (Shoham 1993).

The Cooperation Protocol

To illustrate how the system works, we will consider an example of interaction among two robots. Let be ***A*** and ***B*** the robots specialized in the exploration of the labyrinth and ***Sup*** be the supervisor. There are three cases of interaction between agents:

(i) A specialist is available and there is an area which is not explored. The robot will begin the exploration of the new area.

```
A: inform(Sup, available_agent(A));
Sup:  request(A, explore(P));
```

P is a point from where the robot A will start the exploration process. The point P is identified by the supervisor ***Sup***.

(ii) A specialist is available and there is not a new area to explore. In this case, the available robot will cooperate with the more loaded one.

```
A: inform(Sup, available_agent(A));
Sup:  inform(A, acquaintance(B));
```

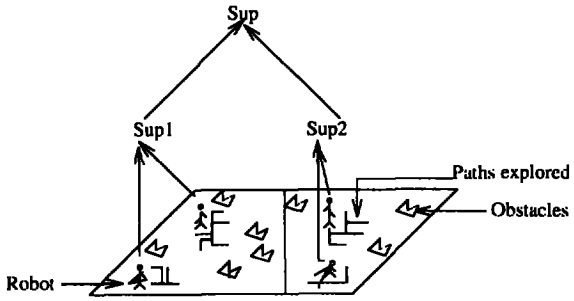

Figure 1: The application environment

```
A: request(B, cooperate(A));
B: inform(A, L);
```

(iii) A robot is more loaded, and the supervisor will search a least loaded one. Thus, the more loaded robot submits to the least one a list of alternative points to explore.

```
A: request(Sup, loaded_robot(A));
Sup:  inform(B, acquaintance(A));
B: request(A, cooperate(B));
A: inform(B, L));
```

L is a list of alternative points.

Another aspect addressed in this application is the load balancing issue. A robot is considered as loaded if the size of its alternative point stack exceeds a given number. All robots which have the size of their stack superior to the fixed number, must inform periodically their supervisor about their load. In the application, the size of the stack is fixed to 50. The latter number is fixed randomly and can be changed until we obtain a best efficiency of the system. Thus, the supervisor has a global vision of the load of its robots, and then it can determine instantaneously the more loaded robot and the least one. An example of execution is given via the Fig.2.

In the case where the surface of the labyrinth is very important, it is more efficient to decompose the labyrinth into several blackboards and to use many supervisor agents (see Fig.1). Each supervisor holds its own blackboard which serves as a current solution for the group of robots. Supervisors of groups are supervised by a global one which has a global vision of the blackboard. Its role consists in decomposing the blackboard in several regions and affects them to the supervisors of groups. It allows also to build the global solution by making the intersection between different blackboards of the supervisors of groups. It has also a representation of different areas which must be explored. Thus, for each area, it holds coordinates of an area-entry point. A robot can start exploration in one area and achieves it at another one supervised by another supervisor agent. In this situation, it detects the limit of the area change

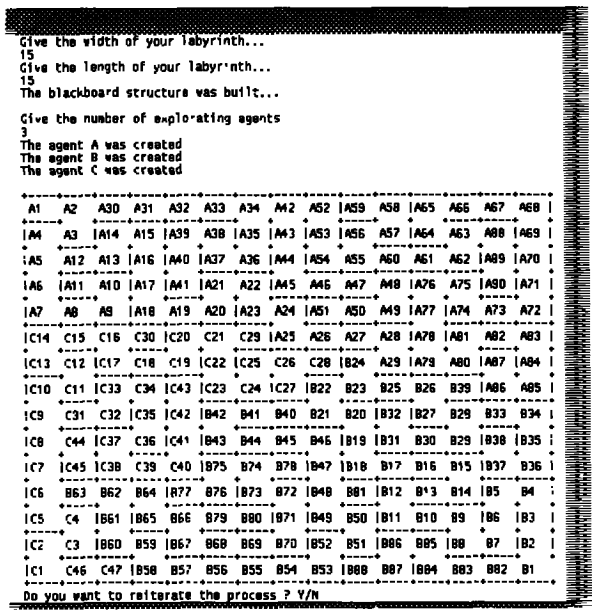

Figure 2: Cooperation of three robots

and then begins interactions with the supervisor that controls the new area. Of course the efficiency of the system is ensured only if agents are distributed as autonomous processes on different machines.

Robots are endowed with multiples exploration capabilities. The application comprises three kinds of group supervisors. The first one coordinates interactions of robots that explore the labyrinth vertically; i.e. when a robot reaches a square from where there are two possible directions (vertical and horizontal), it chooses to progress in the vertical one (see Fig.3). The second one supervises robots exploring the labyrinth horizontally; i.e. when arriving to the intersection square, a robot prefers to progress in the horizontal direction. Finally, the third supervisor of group manages a group of robots that do not backtrack when arriving to an intersection of two paths. Furthermore, Each robot is capable to adapt itself to leave its group and to rejoin another one. After joining a new group the robot must load a new capability corresponding to the kind of the exploration of the group.

Implementation

The application is implemented for a Unix-stations network under a programming environment called PM^2 (Parallel Multithreaded Machine) (Namyst & Mehaut 1995). The latter offers the possibility for agents distributed on different machines to communicate directly by message-passing. It provides a transparent model of communication between threads located on different machines. The communication under PM^2 is based on one primitive called LRPC[2]. In our case, a robot agent

[2]Lightweight Remote Procedure Call

corresponds to a slave process, whereas the supervisor agent corresponds to a master one. The application can be viewed as a set of threads communicating with each other via a work-stations network. Each agent is implemented as a concurrent object containing a set of threads. The number of agents can change. The supervisor has the capability either to add or to remove dynamically a robot. An agent's capability is programmed either as an LRPC service or as a method. An LRPC service is a piece of code which can be invoked by several threads in parallel. The latter can be either a local thread or a remote one. The blackboard is a structure shared by all robots. It is represented by a file. To solve the problem of consistency due to multiple accesses to the blackboard, we have implemented a mechanism of mutual exclusion based on a semaphore. This mechanism is composed of two LRPC services named respectively: ***blackboard_access*** and ***blackboard_leave***. In addition to previous LRPC services, a robot has a capability called ***explore*** which allows him to start the exploration process. In the blackboard, a path is represented as a list of structures containing coordinates of points. For the robot, the cooperation protocol is supported by the LRPC services called respectively: ***acquaintance*** and ***cooperate***. The LRPC service ***acquaintance*** is invoked from the supervisor which requests an available robot and provides him its acquaintance. The latter represents a loaded robot. The LRPC service ***cooperate*** is invoked by the available robot, and the thread correspondent is performed at the loaded one. This latter provides a list of alternative points to explore. For the supervisor, the cooperation protocol is supported by two LRPC services: ***available_robot*** and ***loaded_robot***. The LRPC service ***available_robot*** is invoked by an available robot, whereas ***loaded_robot*** is invoked by a loaded one. Furthermore, the supervisor is characterized by two capabilities, ***activate_robot*** and ***initialize_environment***, programmed as methods. The capability ***initialize_environment*** enables the supervisor to generate a new labyrinth, whereas ***activate_robot*** allows the him to create a new robot agent, and asks him to start exploration. The robot has an attribute ***stack_alt_point*** having the structure of a stack. Whenever a robot progresses in its exploration process it pushes all alternative points at its stack. Whenever a robot has explored a path, it pops its stack and reiterates the process. When the stack is empty the robot requests its supervisor for cooperation. The supervisor has two data structures: ***available_table*** and ***loaded_robot***. The structure ***available_table*** holds the list of available robots, whereas ***loaded_robot*** holds the load of robots.

Advantages of the System

In this section we discuss advantages of our architecture.

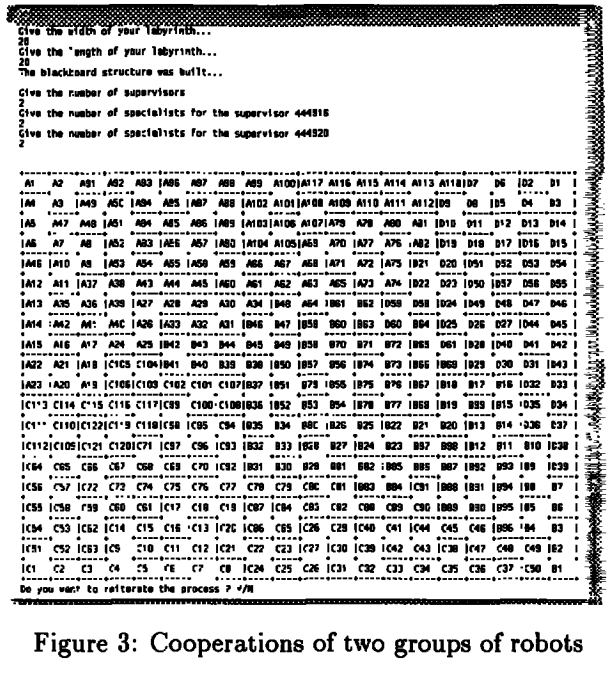

Figure 3: Cooperations of two groups of robots

- Since solutions explored by robots are hold at the blackboard, and robots are endowed with complex capabilities of communication and reasoning, MAGIQUE allows to implement easily the application.
- The control of the system is more efficient because it is hierarchical and is based on many supervisor agents. The control knowledge as the load of each robot and their state (active or idle) is represented explicitly in supervisor agents.
- Each robot can be considered as an autonomous agent because it has capabilities for acting (moving in the labyrinth), perceiving its environment (it can detect if its neighbour points are obstacle or no), communicating with its acquaintances (robots cooperate with each other to solve the load balancing issue) and making decisions (a robot informs its supervisor about its load rate, requests its acquaintance to obtain points of cooperation).

Related Work

SIM_AGENT (Sloman & Poli 1995) is an experiment toolkit intended to support exploration of design options for one or more agents interacting in discrete time. The system is used to implement a simulated trainable robot called RIB (Robot In a Box), which has a hybrid architecture based on the 'Filter' conditions and 'Select' actions. In (Balch 1997), Tucker Balch describes research investigating behavioral specialization in learning robot teams. The experiments, conducted in robot soccer simulations, evaluate the agents in terms of performance, policy convergence, and behavioral diversity. In (Tambe & Rosenbloom 1995) an architecture for agents that track other agents in

multi-agent world is implemented as an experimental variant of the SOAR integrated architecture, that conforms to a set of requirements specific to the described application. Agents based on this architecture have been implemented to execute two different tasks in a real-time, dynamic, multi-agent domain. The architecture of the electronic market and the behaviours of the agents are presented. The system DVMT (Distributed Vehicle Monitoring Testbed) is described in (Lesser & Corkill 1983). It simulates a network of vehicle monitoring *nodes* (agents), where each node is a problem solver that analyzes acoustically sensed data to identify, locate, and track patterns of vehicles moving through a two dimensional space. Each problem solver has a blackboard architecture with blackboard levels and knowledge sources appropriate for vehicle monitoring. The CONTRACT NET model (Davis & Smith 1983) is similar to our model concerning the architecture aspect. The structure of CONTRACT NET is hierarchical and the protocol consists in negotiating contracts between a manager agent and a group of agents specialized in solving the task submitted for them by the manager. In this protocol, agents do not know either how the other ones will achieve the submitted task, nor how the manager will exploit the results replied by the agent which has performed the task.

Conclusions and Future Work

Through this article, we have proposed an instantiation of the model MAGIQUE to simulate a group of robots, represented as autonomous agents, in the exploration of a labyrinth. The implemented system illustrates a group of autonomous agents interacting either directly with each other or indirectly via their supervisor. The application is an example demonstrating that the model of autonomous agent and the model of blackboard can be combined to make up a robust one. The blackboard structure contains explored paths and represents the current solution. The implementation of the system has revealed that for systems which are characterized by a global solution, MAGIQUE is an interesting modeling for supporting these solutions. Global information, related to the load of agents and their state, are represented explicitly at the supervisor. We have noticed that in the case where a labyrinth has small dimensions and there are several robots participating in the exploration process, the performance of the system is deteriorated because there are an important density of communication between agents. An issue we are interested to consists in establishing a relationship between the surface of the labyrinth and the number of active robots in order to improve the efficiency of the system. We are attempting also to determine dynamically the number of supervisors which will coordinate interactions of robots regarding to the dimensions of the labyrinth. Each supervisor of the group supervises a group of robots exploring a part of the labyrinth, and a global supervisor will manage interactions of supervisors of groups. The aim is to distribute the coordination process dynamically through a set of supervisors of group in order to improve the performance of the system. After the implementation of the application on a Unix-work station network, we will implement it on other types of architectures like DEC/ALPHA processors, and compare results to know what architecture is adapted for.

References

Balch, T. 1997. Learning Roles: Behavioral Diversity in Robots Teams. In *AAAI Workshop on Multiagent Learning.*

Bensaid, N., and Mathieu, P. 1997. A Hybrid and Hierarchical Multi-Agent Architecture Model. In *Proceedings of the Second International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agent Technology, PAAM97,* 145 155. London-United-Kingdom: The Practical Application Company Ltd.

Cohen, P. R., and Levesque, H. J. 1990. Intention is Choice with Commitment. *Artificial Intelligence* 42:213–261.

Davis, R., and Smith, R. G. 1983. Negociation as a Metaphor for Problem Solving. *Artificial Intelligence* 20 n° 01.

Lesser, V. R., and Corkill, D. D. 1983. The Distributed Vehicle Monitoring Testbed : A Tool for Investigating Distributed Problem Solving Networks. *AI Magazine* 4(3):15-33.

Muller, J. P., and Pischel, M. 1993. The Agent Architecture InteRRaP: Concept and Application. Technical Report 93-26, German Artificial Intelligence Research Center (DFKI), Saarbrucken. Research report.

Namyst, R., and Mehaut, J.-F. 1995. PM^2: Parallel Multithreaded Machine. A computing environment for distributed architectures. In *ParCo'95 (PARallel COmputing).* 179–184.

Shoham, Y. 1993. Agent-Oriented Programming. *Artificial Intelligence* 60:51 92.

Sloman, A., and Poli, R. 1995. SIM_AGENT: A toolkit for exploring agent designs. In *Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages,* 292 407.

Tambe, M., and Rosenbloom, P. S. 1995. Architectures for Agents that Track Other Agents in Multi-Agent Worlds. In *Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages,* 156 170.

Wooldridge, M. 1995. Time, Knowledge, and Choice. In *Proceedings of the 1995 Workshop on Intelligent Agents: Agent Theories, Architectures, and Languages,* 79–96.