An experimental approach to cooperative learning of multi-agent systems

Hitoshi MATSUBARA, Kazuo HIRAKI, Yoichi MOTOMURA, Yasuo KUNIYOSHI,
Isao HARA and Hideki ASOH
Electrotechnical Laboratory
Tsukuba, Ibaraki, JAPAN 305
e-mail: matsubar@etl.go.jp

Abstract

We have built a multiple autonomous-robots simulator in order to study cooperative learning of multi-agent systems. By using the simulator, experiments can be done for any tasks by any learning methods with some conditions. The simulator makes it possible to approach empirically to cooperative learning.

'Learning of multi- agent systems' has not been founded yet as a field of study. Certain points have not been made clear: what is learning of multi-agent systems?; what questions should be raised?; what should be achieved; what should be considered as the cost (for example, should the cost of communication between agents be considered?). We intend to found a field, 'learning of multi-agent systems', through the experiment of a concrete example of appropriate complicatedness.

1 Introduction

We have built a multiple autonomous-robots simulator in order to study cooperative learning of multi-agent systems. By using the simulator, experiments can be done for any tasks by any learning methods with some conditions. There have been only few researches that have dealt with the cooperative learning of multi-agents in realistic domains. The simulator which is introduced here, however, makes it possible to approach empirically to cooperative learning.

In a multi-agent system, a task is carried out by cooperation among agents. Desirable types of cooperation varies according to the complicatedness of the outer world other than agents, the ability of the agents, the quality and quantity of the task, and dynamic change of the environment which includes the agents and the task. The importance of learning of multi-agent systems has often been pointed out, and some studies have dealt with the matter. But they are limited to simple domain setting. 'Learning of multi- agent systems' has not been founded yet as a field of study. After all, certain points have not been made clear: what is learning of multi-agent systems?; what questions should be raised?; what should be achieved; what should be considered as the cost (for example, should the cost of communication between agents be considered?). We intend to found a field, 'learning of multi-agent systems', through the experiment of a concrete example of appropriate complicatedness.

2 A Multiple Autonomous Robot Simulator: MARS

An example which is chosen as the subject that has appropriate complicatedness is the case in which multiple behavior-based autonomous robots cooperate to carry out a task in a physical environment. The final goal of the study should be realized on real robots, but the operation of cooperative task on real robots alone still has many problems to deal with. Therefore in this experiment a simulator is built, on which an experimental environment for cooperative learning is prepared. The simulator is extended from MARS(Multiple Autonomous Robot Simulator)[1], which is now under development for robotic researches, in order to apply for the experiments on cooperative learning. The simulator which has been made simplifies the real world, but it is planned practically on a realistic bases. The architecture of MARS is based on *Subsumption*

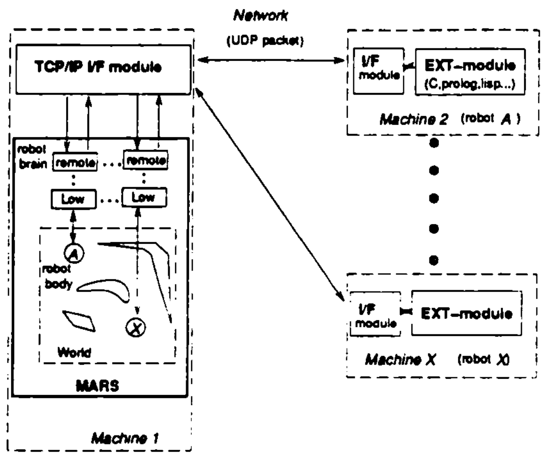


Figure 1: Current model of MARS

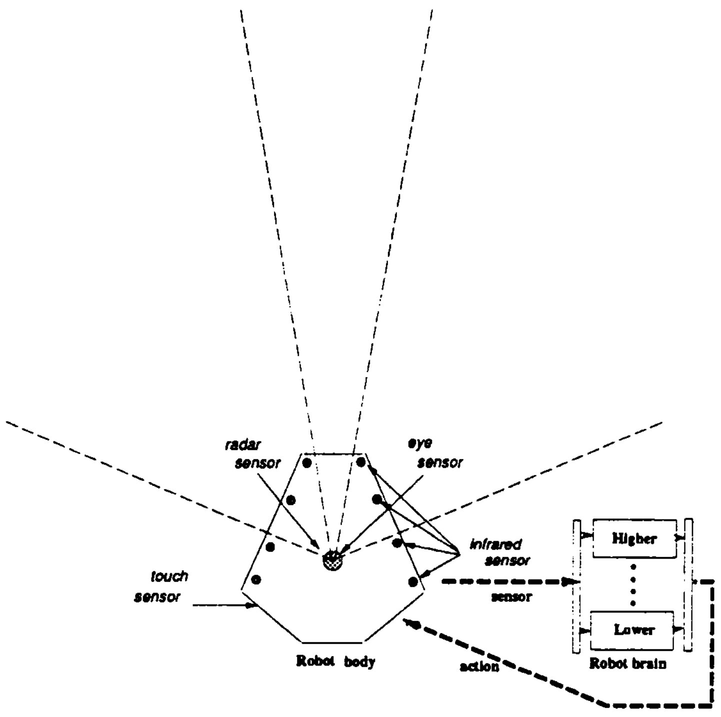


Figure 2: A simulated robot in MARS

Architecture[2] proposed by R. Brooks. The current model is shown in Fig.1. For this study there have been made some improvements to the original model of MARS. A simulated robot is designed to have the body and the brain separated conceptually. The framework, which was proposed by M. Inaba, is called *Remote-Brained robotics*[3]. The body is the main constituent for taking action, and it acts in the physical environment. With its multiple sensors, the body is capable of getting information from physical environment. The brain is designed to control the action by receiving sensor information from the body. The brain consists of stratified control modules. A simulated robot (see Fig.2) can be made by using GUI(Graphics User Interface) in MARS. The simulator is written on EUSLISP[4], which is an object-oriented LISP language for robotics researches.

In the simulator, the sub-control modules are prepared, such as 1. random walk, 2. avoidance of obstacles, 3. escape from deadlock. They can be incorporated when the robots are defined. The sub-control modules are in charge of controls in the lower level (such as a conditioned reflex), and they are brought to action when decision is not made in the higher level. Therefore the sub-control modules are subsumed under the higher modules, which are in charge of advanced and broad decision- making. In order to evaluate the experiment on cooperative learning by various approaches or models, the higher control modules are prefered to be able to be written with any programming languages independently of the simulator. In order to connect this EXT- modules to the simulator, we have developed interface modules. This has made possible the distributed processing through the network by TCP/IP datagram-typed socket. The higher decision-making agent for each of the robots can be operated by different workstations. Since cooperative learning needs a number of complicated and large-scale EXT-modules, this interface system can distribute and reduce the load of calculation on the simulator.

The EXT-module are connected to the interface module by UNIX standard I/O, so we can use any programming languages. The EXT-modules are written to the following format.

```
OUTPUT .... Robot_No (Action list)
INPUT .... Robot_No Time (Value of Sensorl)
                Value of Sensor2 .....
```

Robot_No (an integer number) is an identifier of a robot, which is decided at the time of connection (it can be known from the sensor-input).

Time is a value of time counter in MARS.

OUTPUT:

Action list is a sequence of the following action commands.

- (:spd x) ..... Move forward x in a unit time.
- (:avel x) ..... Change direction for X degrees in a unit time.
- (:nop) ..... Do nothing specifically (stop).
- (:press x) ..... Press an object with strength x.
- (:grasp X) ..... Grasp an object X.
- (:release) ..... Release a grasped object.
- (:send-sensor x (:rotate y)) ..... Rotate a loaded sensor x for y degrees.

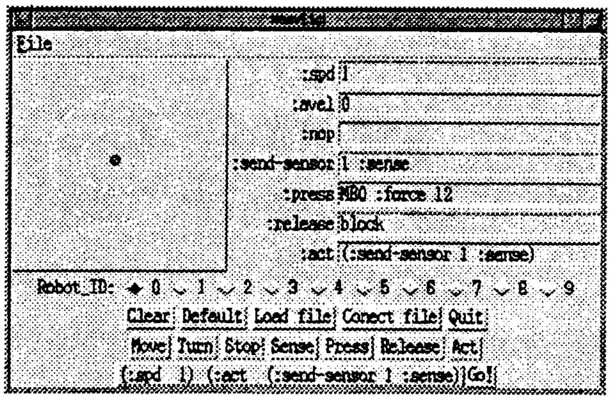
INPUT:

Value of sensor ...... values returned from the following sensors.

- eye-sensor .... Return the label list of object in sight (scope angle *a* degree; distance *L*). This sensor can rotate.
- radar-sensor .... Return the list of distance toward the nearest object on the X (number) radiating lines.
- touch-sensor .... Making touch judgment, return the list of touching points (local coordinate values).
- infrared-sensor .... A infrared sensor. Return t or nil whether there is an object in the specified scope.
- reward-sensor .... Return the value of evaluation (value of a real number decided by a treated object) of an action to treat a task (press:).

In addition, the moving speed of a robot and so on can be obtained.

The EXT-modules function in the following way:


File
Robot_ID: ◆ 0 1 2 3 4 5 6 7 8 9
Clear Default Load file Conect file Quit
Move Turn Stop Sense Press Release Act
Go!

```
:spd 1
:avel 0
:nop
:send-sensor 1 :sense
:press NBO :force 12
:release block
:act (:send-sensor 1 :sense)
(:spd 1) (:act (:send-sensor 1 :sense))
```

Figure 3: Operation panel for teaching of MARS

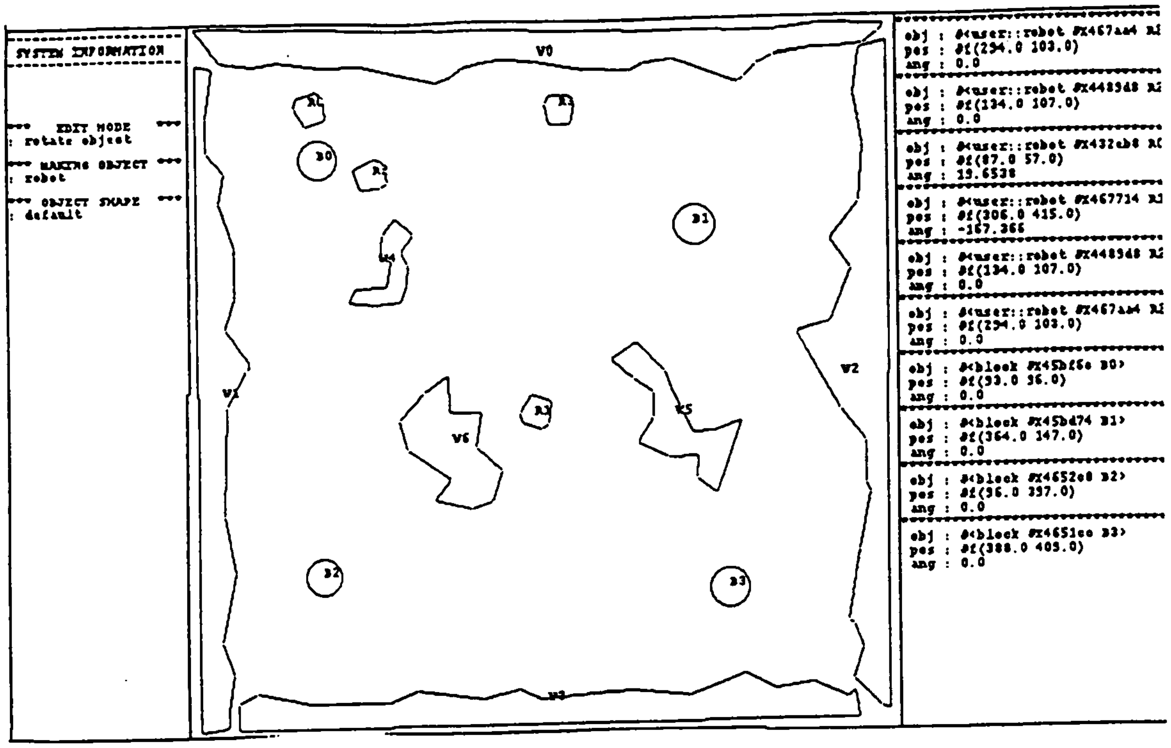
judge the situation by the sensor information obtained from the standard input; do decision-making; select an action; write into the standard output. If, at this point, kind of reward is arranged to be offered toward the action, learning may be realized.

Learning with a teacher is also possible by receiving the action list and the sensor input, if a researcher navigates by an operation panel on X-Window (see Fig.3). This system is effective for learning of all the EXT-modules (with a teacher) by a stochastic model or a neural network model .

As an experimental task, we have selected a *Cleanup Room Problem.* There are a lot of heavy or light objects in the room. There are several robots in the room. Some robots can pick up heavy objects, what others cannot. The robots have to pick up as many objects as they can within a limited time. If a robot cannot pick up an object alone, tow or three robots may pick up the object by cooperation. After cleaning up an object, the robots receive rewards according to weight of the object. The goal for the robots is to receive as many rewards as possible in the limited time. An example of initial setup of this task on MARS is illustrated in Fig.4.

3 An Example of Cooperation of Multiple Autonomous Robots

As for the task mentioned in the last section, multiple robots carry about the objects by cooperation following the program by a researcher. The Fig.5 shows the case in which a researcher makes a program in



```
SYSTEM INFORMATION

--- EDIT MODE ---
: rotate object
--- MAKING OBJECT ---
: robot
--- OBJECT SHAPE ---
: default
```

```
obj : #<user::robot #x467aa4 R
pos : #((294.0 103.0)
ang : 0.0
obj : #<user::robot #x4483d8 R
pos : #((134.0 107.0)
ang : 0.0
obj : #<user::robot #x432eb8 R
pos : #((87.0 57.0)
ang : 19.6538
obj : #<user::robot #x467714 R
pos : #((306.0 415.0)
ang : -167.366
obj : #<user::robot #x4483d8 R
pos : #((134.0 107.0)
ang : 0.0
obj : #<user::robot #x467aa4 R
pos : #((294.0 103.0)
ang : 0.0
obj : #<block #x45bf6e B0>
pos : #((93.0 36.0)
ang : 0.0
obj : #<block #x45bd74 B1>
pos : #((364.0 147.0)
ang : 0.0
obj : #<block #x4652e8 B2>
pos : #((96.0 397.0)
ang : 0.0
obj : #<block #x4651ee B3>
pos : #((388.0 405.0)
ang : 0.0
```

Figure 4: Cleanup Room Problem

advance to let robots cooperate, but the goal of our experiments is to make robots cooperate in various ways through the process of learning.

4 Toward Cooperative Learning

Through the experiment, we are investigating what kind of *"learning by multi-agents"* is possible. The main points of investigation are as follows.

- Sharing and distribution of knowledge

  Learning is supposed to develop by some knowledge being shared by all agents sharing and other knowledge being possessed by individual agents distribution. Then where shall the boundary be formed between sharing and distribution?

- Sharing of satisfaction

  Not only the evaluation of achievement of its own task, learning of each agent must reflects the achievement of all agents as a whole. How could satisfaction be shared among agents?

- Redundancy and robustness

  Each agent develops its individuality through learning. So it is possible to have a *lazy* agent which does not tend to carry out a task. This agent, however, may show its ability in abnormal situations. How are the redundancy and the robustness brought about through learning?

- Learning methods

  Which methods of learning are valuable as candidates for multi-agents systems? For example, is reinforcement learning valid? Further, what should get feedback as a basis toward learning?

5 Concluding Remarks

In this paper we have dealt with a simulator as a tool for studying cooperative learning. The characteristics of the simulator includes: for multiple robots; convenience due to an interface by standard I/O (it can be written in any languages can be used from other machines through a network); wide use as a tool for cooperative learning (any task or any learning method can be dealt with). This simulator is to be PDS software to the public after making further

```
%toplevel

       loop :- sense(DATA),          % sensing
       plan(DATA, Plan),     % planning
       act(Plan),!,          % acting
       add_hist(DATA,Plan),  % add history of Data and plan
       loop.
       loop :- loop.    % In case failing in sensing and/or planning

       %--- planning part !!---

       plan(SenseData, Action) :-
(
   touchable_mb(SenseData) ->    % If touchable to a MB(target)
   grasp_mb(SenseData, Action)   % grasp the MB
        ;
      exist_m_with_r(SenseData) ->  % If there is a robot near a MB
   go_to_m_w_r(SenseData,Action) % go to the MB
        ;
   exist_mb_obj(SenseData)  ->    % If there is a MB
   turn_to_MB(SenseData, Action)  % turn to the MB
;
   exist_obstacle(SenseData) ->    % If there is an obstacle
   avoid_obj(SenseData, Action)    % avoid it
        ;
   exist_ob(SenseData) ->          % If there is an object far away
   turn_to_obj(SenseData, Action)  % turn to the object
        ;
   random_walk(Action)     % Default Behavior(random walk)
        ).
```

Figure 5: An example program for cooperation

revisions. After this we intend to conduct experiments on cooperative learning in this environment, which is the main subject of the study.

References

[1] Y. Kuniyoshi, Y. Mizuno, and M. Kakikura : Multiple Autonomous Robot Simulator on Euslisp - MARS -, The 12th annual conference on Robotics Society of Japan (1994)

[2] R.Brooks: A robust layered control system for a mobile robot, IEEE Journal of Robotics and Automation, vol.RA-2, no.1,pp14-23 (1986)

[3] M.Inaba: Remote-brained robotics: interfacing AI with real world behaviors, Proc. of ISRR-93 (1993)

[4] T.Matsui: Euslisp Reference Manual, ETL technical report, ETL-TR-92-35, Electrotechnical Laboratory, Japan (1992)