# Study on Multi-Agent Simulation System based on Reinforcement Learning Algorithm

Shu Da Wang[1], Shuo Ning Wang[1], Wei Ping Zhang[2]
*1 College of Computer and Information Engineering, Harbin University of Commerce, Harbin, China*
*2 Heilongjiang Province Cereals, Oils & Foodstuffs Import and Export (Group) Co., Harbin, China*
*wangshd@hrbcu.edu.cn*

## Abstract

*Multi-agent simulation system based on reinforcement learning algorithm is a micro-individual acts of modeling and simulation methods, which have wide applicability, distribution, intelligent and interactive features etc. Firstly, studying on reinforcement learning algorithm, and then analysis and design the multi-agent simulation system structure, multi-agent system main modules, the implementation of the definition and finally, carefully design the multi-agent simulation system software, and multi-agent simulation collective system simulation and surrounded the location gathered from the space simulation experiment, the results showed that: Construct a multi-agent simulation system based on reinforcement learning algorithm, achieve real-time simulation of multi-agene, and multi-agent to get effect quickly ,and to quickly construct surrounded conduct by mobile groups, the conduct of the system to achieve the global optimum effect.*

## 1. Introduction

Reinforcement learning algorithm [1] is actually Markov decision-making process. It is a hypothetical, believed that the future status of the agent will rely solely on the current state, and unrelated to the previous state. But agent system environment is very complex, can not obtain state of fully accurate and complete in the current environment, assumptions that the convergence of the algorithm in the actual environment are not met. The purpose of the reinforcement learning is to find the mapping optimization, in state space and in movement space. The agent in the reality environment has a group of movements and the state of discrete and continuous, all of these will be described, it will format combinatorial explosion of the traditional Reinforcement Learning (RL) method[2].

In the reality, the state of the environment and the inside is asynchronous change. Thus noise is very serious, can not built an interactive process modeling by a simple probability distribution .In the actual environment, agent guides learning by state action, but the feedback obtained often is not immediate, and need an uncertain period. This requires learning in real time constraints, increased the difficulty of learning. In the distribution of roles, under certain conditions, the evaluation of the results is a difficult problem. Therefore, design the software of study on multi-agent simulation system is very important based on reinforcement learning algorithm.

## 2. Reinforcement Learning

### 2.1. Multi-agent Reinforcement Learning

Learning is the agent through interactive and the environment to adjustment the observed action mapping. As agent is the purpose by the study to adapt the mapping from perception to action, so improve agent's ability to achieve its objectives. All the MR components by three parts [3]: perception, reasoning and action. Perception depends on the environment, and action will affect the environment. By the environment without losing the general assumption that the agent should the effective reasoning for the chosen expectations.

RL refers to study mapping from the state of the environment to the movement, to make the movement obtain the greatest reward of accumulated value. This method is different from supervised learning technology, by cases of counter-examples to inform taken, but through trial and error to find optimal strategies. agent framework structure [4]of RL as shown in Figure 1.

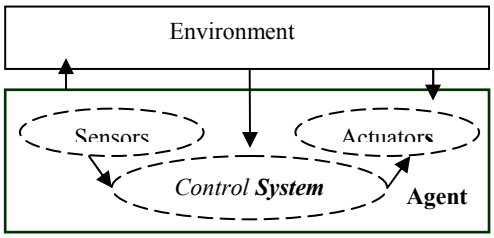



**Figure 1. Framework of RL**

Where, agent implements the perception、action and

978-0-7695-3507-4/08 $25.00 © 2008 IEEE
DOI 10.1109/CSIE.2009.234

control systems modules. Agent observed state of the environment s by perception and transmitted to the control system, Control system decides to take action a and transfer a to the implementation of a given action. Action to change the environment, the environment will be rewarded or punished r return to the agent.

### 2.2. Multi-agent RL Algorithm

Learning algorithm [5] is an important milestone in the RL, Watkins puts forward in 1989, People generally think that the timing difference algorithm and Q-learning algorithm are RL algorithm, however, renowned scholars Sutton from the field of RL that Q-learning algorithm is one of the timing difference, called for the strategy timing difference algorithm (off policy TD). Since Q learning estimates value of the state's action, in order to achieve the strategy, and TD algorithm only to the estimated value of the state are quite different. In the algorithm, tectonic value function of movements and a state is Q function. Q function is defined as follows:

$$Q^{z}(s,a) = \sum_{s'} P_{ss'}^{a}[R_{ss'}^{a} + \gamma V^{z}(s')] \quad (1)$$

Theory prove, when a learning rate a to meet certain conditions, the Q-learning algorithm converges to an inevitable move to a state of optimal value function. The amended formula of Q value is define as following:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max Q(s',a') - Q(s,a)] \quad (2)$$

Q-Learning algorithm is as follows:
Initialize Q(s, a) arbitrarily
Repeat (for each episode)
Initialize s
Repeat (for each step of episode)
Choose a from s using policy derived from Q (e.g., ε -greedy)
Take action a, observer r, s′

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max Q(s',a') - Q(s,a)]$$

$s \leftarrow s'$

Until *S* is terminal. And the *Q* value operates as strategy, more convenient, because only based on the current state, Choose to make a value of *Q (s, a)* the largest movements; and the value function of TD will need travel the next step, all may choose the state of the greatest value *V(s)* as the direction of the implementation movement. The Q-learning model is a non-enhanced learning methods, it moves through the state expect to return mapping action-evaluation function *Q* to resolve non-Markov problem or incomplete information. Therefore directly applied to the Q-learning multi-agent system (MAS) [6] still has some difficulties[7], in order to overcome this problem, Q-learning methods will be improved and combined with other algorithms, the agent used in the study of reinforcement learning.

## 3. Multi-agent Simulation System

### 3.1. Structure of Multi-agent System

Multi-agent simulation system mainly constituted by eight components as follows the communications module, information processing module, enhanced learning modules, individual acts of choice module, select groups of modules, action choices modules, sensor modules and the environment module. Its structure of coordination is shown in Figure 2.

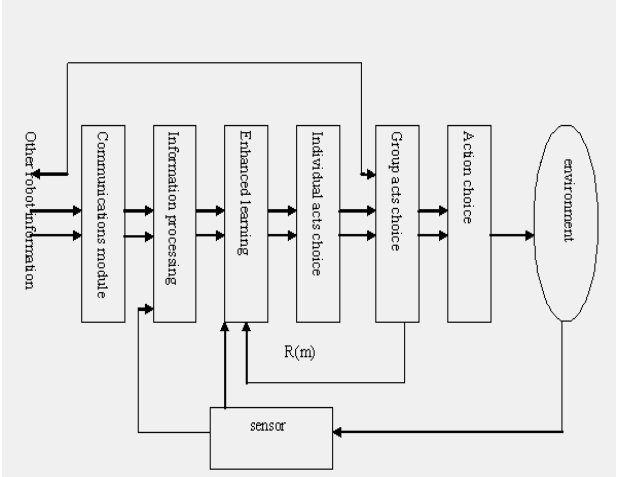



**Figure 2. Coordination acts structure**

The control process of system: First, the agent through the sensor obtains information about the outside world, at the same time communications modules obtain information of others. After passed information processing module information is transfer to enhanced learning modules to plan, and then through individual acts of individual choice modules have decided to act: $(b_1 \ldots b_j \ldots b_M)(b_j \in B)$ ,and the individual choices information of agent through the communications module is delivered to select groups of modules, through the method of election overall action is obtained. According to the overall behavior, agent in the environment implements the appropriate action. According to this system of the environmental movement in the role of the results, strengthen the signal value is given. According to the enhanced signal value, intelligent decides the trend of such acts in this environment. After repeated study, the agent can be constructed independently the mapping

which is from the environment to conduct to ensure the realization of the overall mission.

### 3.2. main modules Multi-agent System

**3.2.1. Multi-agent sensor modules.** Remote sensor is the sensor that are there any other agent in a very large radius (can be seen as an infinite) information within the radio sector[8]. Of such sensors, the return value which sensors j in the sector ε j is expressed as:

$$r_j = \begin{cases} 1, \exists P_i \in \varepsilon_j \\ 0, \forall P_i \notin \varepsilon_j \end{cases} \quad (3)$$

P i said that position of i agent. A group of rj, j = 1,2,…, N, N is a binary string. All possible binary strings provided a R corpora of return value, that is $[\, r_1, r_2, \cdots, r_N \,] \in R$.

Short range sensor is that access information of other agent in R0 radius. Under such circumstances, the radio sector j of the sensor-agent neighborhood said for ε j:

$$\tilde{\varepsilon} \{ P_i \| P_i - P_0 | < R_0 \} \quad (4)$$

P i said that position i agent, P0 said position of sensing agent. The return value that j sensors in the sector ε j can be expressed as follows:

$$r_j = \langle \begin{matrix} 1, \exists P_i \in \tilde{\varepsilon}_j \\ 0, \forall P_i \notin \tilde{\varepsilon}_j \end{matrix}$$

(5)

**3.2.2. Multi-agent information processing model.** Hypothetical system has n agent, each agent has P sensors, each sensor was vague into *q* grades, then the algorithm in the agent's state space is $|q|^{n \times p}$. Each agent has *M* actions, the number of variables of Q function is $|q|^{n \times p} |M|^n$. We can see that the size of the variable space and the number of smart sensors are the number exponentially, with the agent number and the number of sensors increasing, there will be a combination of explosive problem[9]. Q function requirements each state action can be traversed in many times, and the variable index-class traversal problem is NP problem. In the variables space, space action can not be reduced, if to reduce the space is equivalent to reduce the movement of the system in the agent's ability to complete the task can not be guaranteed. In order to ensure the convergence and convergence rate of enhanced learning, the state space must be compressed.

Each binary string which its length is N is returned correspond to the local distribution of environment of a agent. Although there are as many as $2^N$ possible value, but can actually different in the local distribution is a very limited number. If so ring shift (left or right) or anti-ring shift (corresponding algorithm shown below), you can see in every binary string (except 00000000 and 11111111), there is an equivalent of the string. If the two strings of their expression of around sensing agent the approximate distribution are the same, then that is equivalent to two series. Through the implementation of the transfer operation that will obtain a unique local conditions of a group of equivalent binary string corresponding series, known as an incentive. Table 1 gives the length of N number of possible incentives. Definition operator extr, it returned to the sensor sets $\tau$ mapped to inspire the collection $\varphi$:

extr: $\tau \to \varphi$

**Table 1. Possible Incentive data of relative length N**

| N | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Possible Incentive data | 2 | 3 | 4 | 6 | 8 | 14 | 18 | 29 | 42 |

Arithmetic as follows:
#CircularShiftto LeftforM（0<M<N）bits:
NewCodeString（1:（N -M ））=OldCodeString（（M +1）: N）;
NewCodeString（（N -M +1）: N）=OldCodeString（1: M）;
# CircularShiftto RightforM（0 <M <N）bits:
NewCodeString（1: M）=OldCodeString（（N -M +1）: N）;
NewCodeString（（M +1）: N）=OldCodeString（1:（N -M ））;
#Reverse Operation:
fori=1: N
NewCodeString（i）=OldCodeString（N-i+1）;
End

### 3.3. Definition of Acts Executive

Agent groups can be moved in any direction to the any pace. However, in order to facilitate control agent, the agent assumed the direction of the campaign is divided into N sectors, corresponding to N uniform distribution sensor. When N = 8, on the local polar direction of the movement in this direction on the movement known as the

simple acts of the agent. With N-dimensional vector shows the simple act of the agent.

$$[\ Behavior\ ] = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ ..... \\ B_y \end{bmatrix}, B_i \in \{-1,1\} \quad (6)$$

If the vector elements Bi = 1, means that the agent can perform acts i. On the other hand, Bi =- 1 agent can not be said that the implementation of i. Corresponds to a simple agent, further defined a weight vector to express a given incentive, a specific behavioral response after the implementation can be achieved of the probability of success. Weight vector can be written as follows:

$$w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ ... \\ w_x \end{bmatrix} \quad (7)$$

$w_i = -1, if B_i = -1, i = 1,2,...,N, \sum_{i=-1}^{N} w_i \big|_{w_{i\neq -1}} = 1$ need to mention is that every agent in a given time to the actual movement step d0 dependent on the largest mobile step dm and the greatest freedom movement step dp, namely:

$$d_o = \min(d_m, d_p) \quad (8)$$

$d_m$ is confirm in accordance with pre-determined ratio η. That is $\eta = \frac{d_m}{R_0}$ the biggest freely movement step $d_p$ which is decided by the largest displacement of conflict-free from one chosen direction chosen.

# 4. Multi-agent Simulation Software Design

## 4.1. Software Mainly Data Structrue

**4.1.1. Class Eigenvector and Features Matrix.** CLASS is a direct all types of characteristics and parameters of the matrix. The structure described as follows:

$$\begin{bmatrix} i \\ ... \\ i \end{bmatrix} CLASS = \begin{bmatrix} ... \\ C_i \\ ... \end{bmatrix} \leftarrow class_{i,}$$

$$CELL = \begin{bmatrix} ... \\ 1 \times 4 \\ 1 \times 4 \\ ... \end{bmatrix} \frac{\leftarrow class_{i,} cell_j}{\leftarrow class_i, cell_{j+1}}$$

$$HSTORY = \begin{bmatrix} ... \\ H_i \\ ... \end{bmatrix} \leftarrow t$$

$$H_i = \begin{bmatrix} ... \\ 1\times 5 \\ 1\times 5 \\ ... \end{bmatrix}_{M_{CEL\times 5}} \frac{\leftarrow class_{i,} cell_j}{\leftarrow class_i, cell_{j+1}}$$

*Ci* is a sub-matrix of *i* category. In this sub-matrix, the first part is record category; second part is definition parameters; third part is defined goal; such all the other characteristics that define in the part of Ⅳ *Ai*. *Ci* equivalent to the head office of *Bi*, *Gi Ai* with the greatest number of lines are.

**4.1.2. Agent Location Matrix.** CELL defines locations of all types of agent in the environment. As shown in Figure xx in the form here $M_{cell} = \sum_{i=1}^{K} M_i$ , but also for i class section j agent, in CELL the vector is $[i.....j....x_{ij}....y_{ij}]$, here（$x_{ij}$ , $y_{ij}$）is the agent in the experimental environment of the coordinates.

## 4.2. Software Environment

MATLAB is used as a software development environment, high-level language to the platform's call MATLAB using two methods to achieve. One approach is to MATLAB and other applications to establish a client / server relationship, MATLAB as a calculation engine, in other applications call; Another method is to use an independent MATLAB C / C + + functions, such as MATLAB CC + + Math Library, an independent C code, such as fuzzy inference engine. The latter is to use some of the features of MATLAB CC + + to achieve the function, for C / C + + programming call.

## 4.3. Software Design

Agent from the environment to receive the sensor data constitute the agent of incentives and agent based on a variety of conditions on a number of actions to trigger an evaluation study in order to determine their action, through acts of the weight of a Driver Specific incentives

- reaction of contact, trained action patterns will be deposited into the knowledge base, with a group of intelligent choice of the foundation. Multi-agent simulation software's organizational structure[10] is as shown in Figure 3.

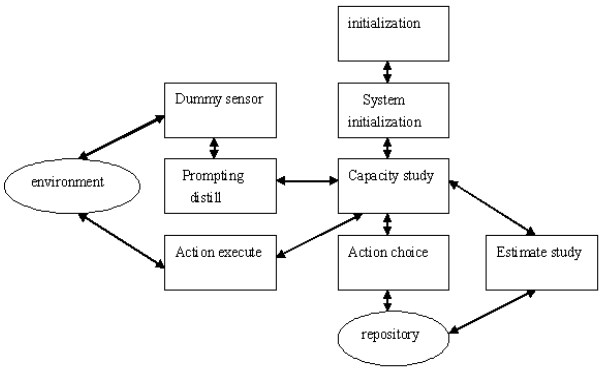
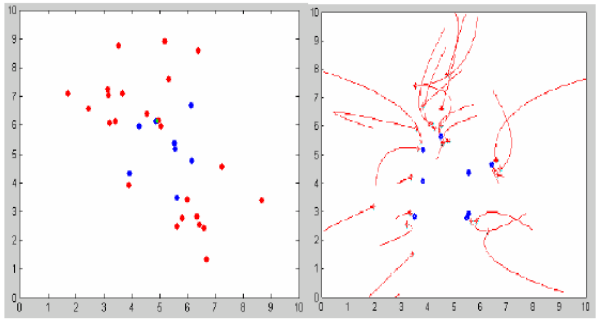



**Figure 3. Software design structure**

## 4.4. Simulation

**4.4.1.Collective siege Simulation.** Simulation results for collective siege are shown in Figure 4. In a 10 × 10 regional, group B (8) and group R (25) agents. Group B agents can perform simple acts, group R agent through multi-agent parallel RL to find the means to collectively search、close tracking and surrounded Group B agent. R group agents use the "*"symbols, Group B agents use "•" Symbol. In the experimental initial state, group B and group R agent random distribution in Figure 4 (a) , With Q study, we find that R group agent can quickly move Group B groups siege, when learning to Step 15 in the circumstances, see Figure 4(b). **It be Can see that all the R group agents are surrounded by B group agents.**



**(a) Initial state** **(b) Learning Step 15**

**Figure 4. DPQ-L algorithm simulation results**

**4.4.2. Location Simulation of Gathered From Space Congregation.** From figure 4(b) can be noted that an R group agent is out of from R group, and then find a new location. The behavior is caused by improvement the Q learning process. It is found that location in step 184 is the best agent finally receives State. Although preceding step 30 has been reached on the state of a local optimum, but at step 177 it chooses a random acts, the acts makes the agent out of the local optimal state. Therefore, before the system reached its optimal state of the overall situation, the smart chooses some of the temporarily poor (or mutation) of the system to a global optimum state are helpful.

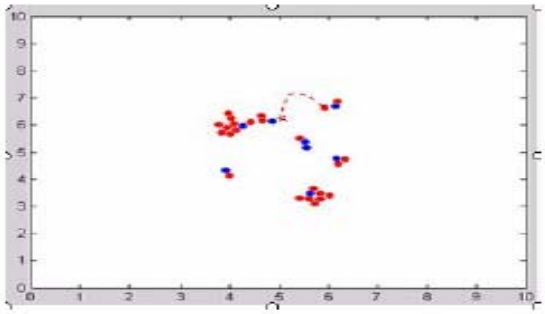


**Figure 5. Location from space**

## Reference


[1] Michael L. Littman, Peter Stone. A polynomial-time Nash equilibrium algorithm for repeated games. Proceedings of 2003 ACM Conference on Electronic Commerce. San Diego, CA, USA, 2003,pp.48-54.

[2] Michael Bowling and Manuela Veloso, Convergence of Gradient Dynamics with a Learning Rate. Artificial Intelligence, 2002, pp.215-250.

[3] Michael L. Littman, Peter Stone. A polynomial-time Nash equilibrium algorithm for repeated games. Proceedings of 2003 ACM Conference on Electronic Commerce. San Diego, CA, USA, 2003,pp.48-54.

[4] Yuko Ishiwaka, Takamasa Sato , Yukinori Kakazu. An approach to the pursuit problem on a heterogeneous multi-agent system using reinforcement learning .Robotics and Autonomous Systems, 2003, pp.245-256.

[5] Brooks，RodneyA. .A Robust Layered Control System for a Mobile Robot.IEEE Journal of Robotics and Automation vol.RA-2, no.1，2006,pp.14-23.

[6] Shun Feng S, Ted T , Hung T H. Credit assigned CMAC and its applicant ion to online learning robust controllers. IEEE Trans on S y stem s, Man and Cybernetics — Part B : Cybernetics, 2003, pp.202-213.

[7] Kitano H, Tambe M, Stone P,et al. The robot-cup synthetic agent challenge 97.RoboCup-97:Robot Soccer World Cup I[C]. Berlin: Springer Verlag,2005.pp.62-73.

[8] Stone P. Layered learning in multi-agent learning .Pittsburgh :Carnegie Mellon University,2004,pp.22-25.

[9] Kaelbling P L, Littman L M, Moore W A. Reinforcement learning: a survey. Journal of Artificial In-telligence, 2004,pp.237-285.

[10] Shuda WANG, Qing ZHU, Zhanqing LUI, Jing YANG, Feng SI. Study of reinforcement learning based on multi-agent robot systems. *Journal of Computational Information Systems* 2007,pp.2001-2006.