![HAL logo](archives-ouvertes.fr)

# Adaptive Mechanism Based on Shared Learning in Multi-agent System

Qingshan Li, Hua Chu, Liang Diao, Lu Wang

**HAL Id: hal-01383324**
**https://hal.inria.fr/hal-01383324**

Submitted on 18 Oct 2016

# Adaptive mechanism based on shared learning in Multi-Agent System

**Qingshan Li\*, Hua Chu\*\*,Liang Diao\*\*\*,Lu Wang\*\*\*\***

*\* Software Engineering Institute, Xidian University,*
*Xi'an 710071, China*
*E-mail: qshli@mail.xidian.edu.cn*

*\*\* Software Engineering Institute, Xidian University,*
*Xi'an 710071, China*
*E-mail: hchu@mail.xidian.edu.cn*

*\*\*\* Software Engineering Institute, Xidian University,*
*Xi'an 710071, China*
*E-mail: diaoliang@stu.xidian.edu.cn*

*\*\*\*\* Software Engineering Institute, Xidian University,*
*Xi'an 710071, China*
*E-mail: Lu_Wang@stu.xidian.edu.cn*

ABSTRACT: *In view of the deployment environment of the adaptive system is complex, dynamic, unpredictable, focusing on the construction of dynamic, uncertain environment adaptive system, and this paper combines the reinforcement learning technology and software agent technology to propose an adaptive mechanism based on shared learning in multiple agent system. Based on this, framework for constructing adaptive systems and shared learning algorithm of agent are given. Finally, by conducting a comparative experiment and result analysis to verify the feasibility of the theory put forward by this article.*

KEYWORDS: *adaptive mechanism; shared learning; Multi-Agent System; system framework*

## 1. Introduction

In the key application areas of software system, system environment is very complex, and the specific performance features are dynamic, changeable and difficult to control [1]. Software systems with self-adaptive capacities can adapt to the changing environment better [2]. In dynamic and uncertain environment, traditional techniques have been difficult to meet the development needs of self-

adaptive systems [3]. In view of the above application requirements and technical challenges, this article uses the following two techniques to support the development of self-adaptive systems: Agent-based technology and reinforcement learning technology.

The second chapter of this paper introduces the self-adaptive system development framework; the third chapter focuses on the description of self-adaptive mechanisms based on shared learning, including self-adaptive processes and shared learning algorithm; the fourth chapter verifies the share learning-based self-adaptive mechanism proposed in this paper, combining case analysis; the fifth chapter summarizes the full text and discusses the further research.

## 2. Supporting framework for self-adaptive system development and running

By referencing FIPA specifications [4], we put forward the support framework of self-adaptive system for development and running as shown in figure 1. The entire frame consists of two core parts: supporting platform and development tools.
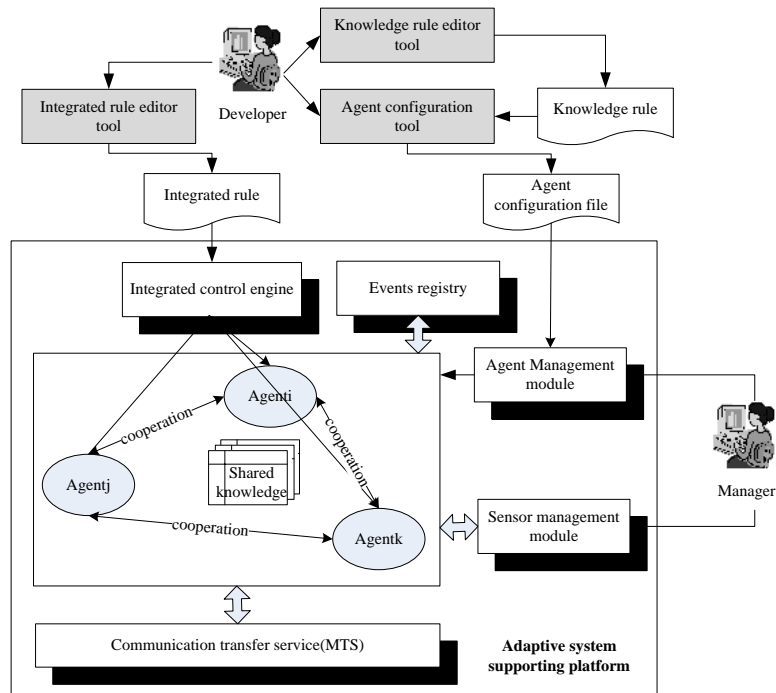


**Figure 1**. *Supporting framework of self-adaptive system for development and running*

The supporting platform provides communication services, event services, and capabilities of the agent lifecycle management, the interaction of distributed multi-agent system and the perception of agent environmental.

Integrated control engine loads the user-specified integrated rules and distributes them to agents, so that the agents can collaborate independently under the guidance of integrated rules. Event registry maintains a list of all events in the system, which agent is interested in, and notifies the agent when the event occurs. Agent library management tool is a special agent in the platform. It is primarily responsible for the static management of local agent and the instantiation of agent.Sensor management module manages the different sensors, therefore the agent can dynamically load different sensors to active perception information of non-active in environmental.

Developers can develop needed configuration files for the system by using self-adaptive system development tools. User also can define collaborative relationships between agents by using a scripting language or graphical way. And user can abstract knowledge rules inside the appropriate agent, using scripting language describe knowledge rules to control the agent to process information automatically, so that the system can complete the decision-making process fast and accurately. Agent configuration tool mainly is used for defining simulation Agent basic information which is required.

## 3. Self-adaptive mechanism based on shared-learning

Reinforcement learning is considered to be one of the core technologies to design the intelligent agent [5]. To cope with unpredictable, dynamic environment in the process, self-adaptive system based on reinforcement learning will interact with the environment continually at runtime, and judge self-adaptive operations performed in a different state to formed knowledge.

### 3.1. *Self-adaptive process based on shared-learning*

In the self-adaptive system, agent is constantly monitoring the environment and sensing the current state of the environment, thus constantly triggering "perception-select-do-learning" process. At the same time, knowledge table is also constantly updated and maintained, which enables the agent to choose actions more responsive to environmental change according to knowledge table, improving effects that agent adapt to environment the gradually.

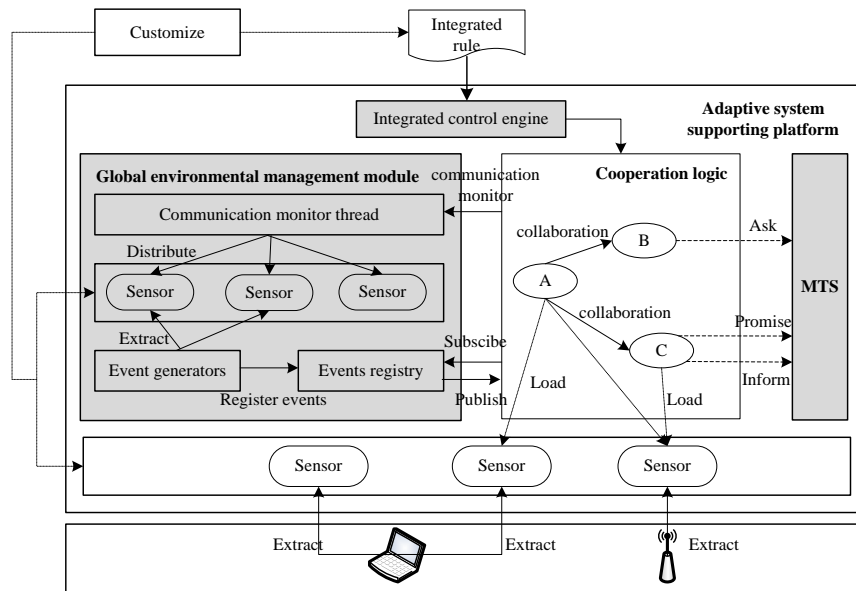Adaptive mechanism based on shared learning in Multi-Agent System



**Figure 2.** *Self-adaptive process based on shared-learning*

Self-adaptive process based on shared-learning is shown in figure 3, the whole process can be divided into the following four steps:

Step1: (sensing stage) Agent interacts with the environment in real time to access the environmental information.

Step2: (decision-making stage) Agent sends the state of the environment (assume that the state is s) to behaviour selecting unit after perceiving environmental change; behaviour selecting unit selects a most appropriate behaviour (assume the action is b) from multiple behaviours which might be performed according to selecting policy and submitted to the execution units.

Step3: (implementation stage) Execution unit executes behaviour directly or executes by calling a number of external interfaces (such as communication interfaces, and so on). Implementation of behaviours may affect the state of the environment;

Step4: (learning stage) Learner is learning unit of self-adaptive agent, Learner has learning algorithm internal. Using state of environment and environment return value of action as entered, learner uses learning algorithm to learn optimal acts policy of self-adaptive agent in environment states, updates Q value that implement acts b at states. When the environment changes, self-adaptive agent calls actions or behaviours that are defined in the agent to adapt to the changes of environment according to knowledge exist in learner.

### 3.2. *Algorithm of shared-learning*

Multi-Agent Collaborative reinforcement learning algorithm proposed in this paper is combined with the rules of knowledge and experience sharing between tuples. The knowledge obtained from each Agent independently learning and convergence will have an appropriate similarity transformation to get a shared experience tuples, which will achieve the solving method sharing about the similar problem. Meanwhile the status and action of the shared experience tuples will be designed as the rules of knowledge and stored in the rule base. While the Multi-Agent system is running, each Agent first query the rule base, match the conditions of rules. If satisfied, then the corresponding actions of the rules matched will be taken. If not, the Agent will turn in the state of the self-learning, get the convergence of the pair of the state and action, then convert it into a shared experience tuples, design rules, finally store this rule into the rule base. A unified space of the behaviour states has achieved by this algorithm through the sharing of the tuples' experience and the rules of knowledge to improve the efficiency of self-learning Agent module.

S: States set is which agent concern, S= {s1, s2 ...};

A: Actions set is defined in agent, A= {a1, a2 ...};

1. Query the rules in the rule base, if the current state is satisfied with the conditions of rules, then the corresponding action will be executed, otherwise the intensive learning will be executed as (2);

2. Initialization: S = s, Q (s, a) = 0, in which $\forall s \in S, \forall a \in A$;

3. Choose an action "a" based on a ε-greedy strategy;

4. the Agent performs an action "a" to get the immediate return "r" and the next state "s'";

5. update Q: $Q(s, a) \Leftarrow A(s, a)+\alpha(r + \gamma \max_a Q(s', a')-Q(s, a))$

   Where $\gamma$ is the discount, $\alpha$ is learning rate.

6. s = s', if all of the Q (s, a) are convergent, executed (7), else (3);

7. reset Agent initial state, repeat 2-6 steps, the learning process will be executed several times until the Q (s, a) is convergent, then the study task could be considered as completed.

8. Each the pair of the state and action obtained by the Agent using the Q-learning will have a similarity transformation to get the shared experience tuple (s *, a *). And the tuple will be designed as rules in which the state is the condition and the behaviour is the action. The rules will be stored into the rule base,

when the conditions of the rules are met, the corresponding action will be executed.

In the initialization part of the algorithm, we build the Q table and initialize the Q value of each state-behaviour to 0. Of course, Q can also be set into a more meaningful value for experienced designers to speed up the convergence of learning. When algorithm get into the loop part, it will execute the process "choice behaviour-implement behaviour-receive return value-watch the new state-update Q table-set up the next loop starting state-choice behaviour ...-update Q table ..." repetitively. When all the values in the table of sharing-experience tuple become convergence, the loop is end.

## 4. Self-adaptive mechanism based on shared-learning

In this chapter, we implement an air defence command and control system to intercept intrusion target. As shown in Figure 3, our three UVAs are flying in the area near the border and decide flight behaviour according to the current location of the intrusion target. Learning trail of shared learning algorithm is the return value from the simulated environment of simulation system and three aircraft shares experience tuple what they have learned.
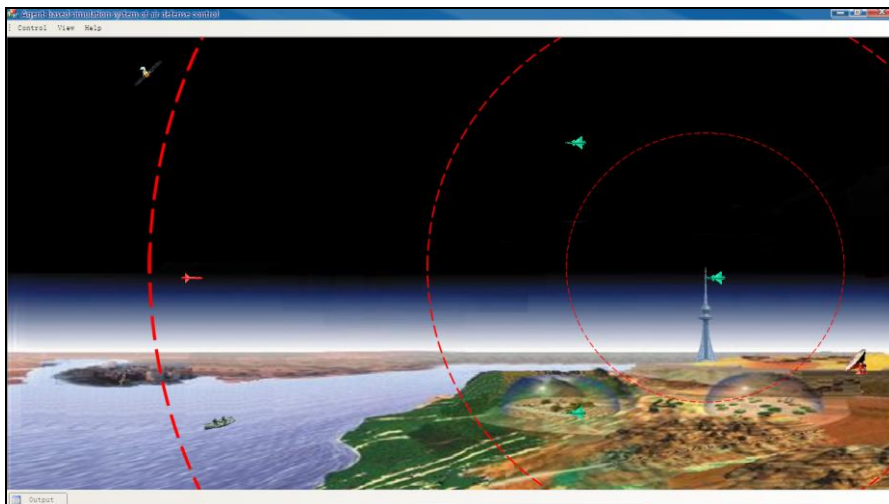


**Figure 3.** *Situation maps of UAV formation intercept invasion target*

### 4.1. *System running*

When agent initialization is completed, each agent receives information from the outside and works in terms of internal knowledge rule file. The generated messages

is sent out though collaborative relationship. Agents finish the blocking process through collaborative interaction. Agent communication processes is shown in Figure 4.
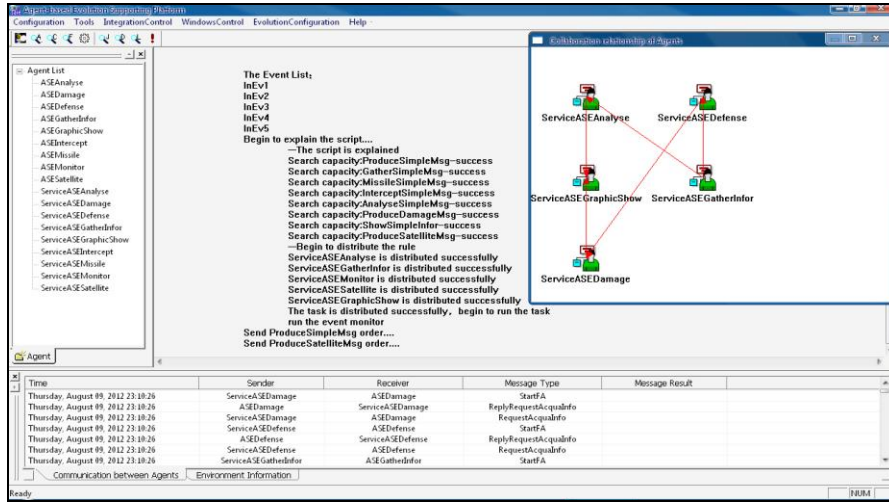


**Figure 4.** *Interface of explain and distribute integrated script*

### 4.2. Results and analysis

Set the appropriate parameters for the problem of interception of UAVs to make shared learning algorithm meet the required conditions. Experimental basic conditions are as follows, a series of successor experiments have been conducted on the basis of change different basic conditions.

(1)The UAV and intrusion target selecting one walking behaviour randomly, behaviour collections are represented as {eastbound, westbound, northbound, southbound, stationary}; (2)The UAV has the ability to learn and make independent decisions respectively, by receiving feedback information of environment to improve its policy; (3)Do not have shared experiences tuple; (4)Assume that initial Q value of each state-behaviour to 0;(5)Instantaneous return functions which environment feedbacks to the UAV are as follows:

**Table 1.** *Return value which environment feedbacks to the UAV*

|  | arrive at the target | the distance to target reduce one step | the distance to target is decreasing | the distance to target keep unchanged | the distance to target increase one step | the distance to target is increased |
|---|---|---|---|---|---|---|
| r | 300 | 100 | 100 | 0 | -1 | -10 |

This section verifies the effectiveness of shared learning and different return values impact of the system learning though two contrast experiment.

Comparative experiment 1 UAVs execute task of blocked, comparison curve of moving times between UAVs which does not use shared-learning and UAVs which use shared-learning.

The result is shown in Figure 5. In the figure, horizontal represents the number of experimental group, each group represents l0 flight simulation, and ordinate represents the moving times of unmanned plane in each set of experiment.
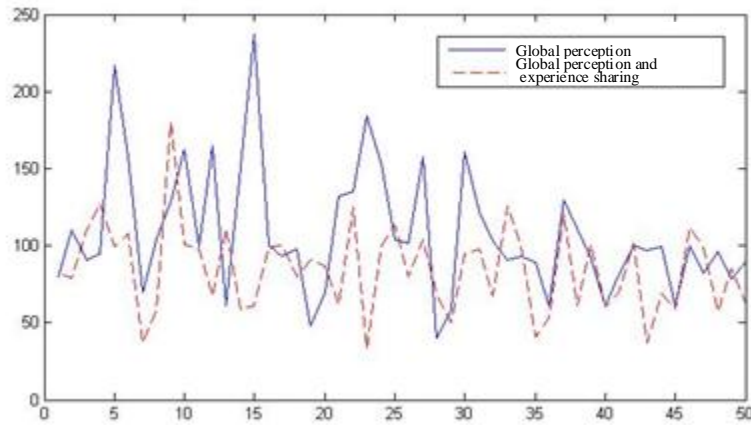


**Figure 5.** *Verification for effectiveness of shared-learning*

In the case of all area perceptions, UAVs can reach agreement on how to intercept invasion targets as soon as possible; learning between the UAVs is the synchronization. After a running process of learning, agent is able to achieve the effect of convergence of expectations.

This experiment will change the basic conditions (3) form not shared experiences tuple to share experiences tuple, each UVA uses a shared learning algorithm for learning. Seen from the experimental results, compared with does not use shared experiences, agent uses shared experiences can convergence to stable values faster, this indicates that the shared learning algorithm to significantly improve the efficiency of collaboration.

Comparative experiment 2 UAVs execute task of blocked, comparison curve of moving times between before change return values and after change return values.

On basis of experiment 1, we change return values set to try to observe the influence of return value on learning. We change return value set of basic condition (5) to:

**Table 2.** *Amendatory return value which environment feedbacks to the UAV*

| | arrive at the target | the distance to target reduce one step | the distance to target is decrescendo | the distance to target increase one step |
|---|---|---|---|---|
| r | 300 | 100 | 100 | -50 |

The result is shown in Figure 6. In the figure, horizontal represents the number of experimental group, each group represents l0 flight simulation, and ordinate represents the moving times of unmanned plane in each set of experiment.
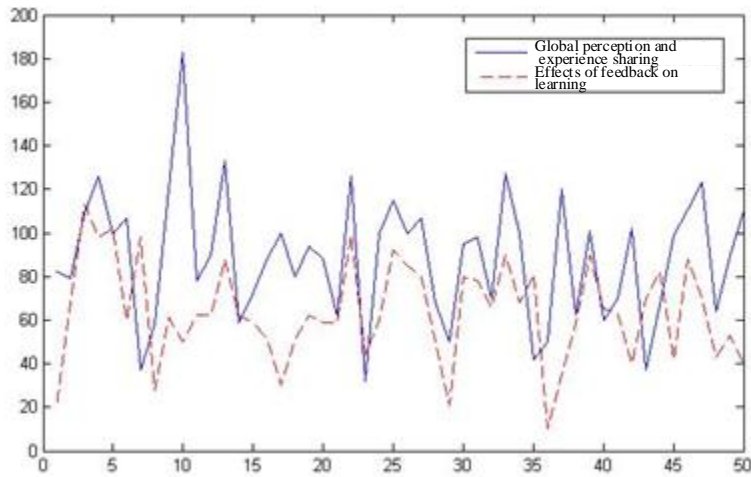


**Figure 6**. *The effect of different feedback value to learning*

Setting return value of increase distance between ion task and target location with a step to-50 make the algorithm get a faster convergence effect.

Through these two experiments, the following conclusions can be obtained: shared-learning algorithm is effective for multi-agent team and come to convergence under certain conditions; Collaborative learning between agents can improve the learning efficiency of multi-agent team. The return value which acts as the only clue of learning, different set of return values can bring to different effect of the system's learning.

**5. Conclusion**

This paper proposes an operation mechanism and learning algorithm for self-adaptive multi-agent system, which makes the self-adaptive agent have the learning capacity and can use the learning algorithm to learn related knowledge of environment online, can develop out effective self-adaptive policy according to the change of environment. All of these enhances the self-adaptive capacity of the system in dynamic and uncertain environment greatly.

This major work of this paper can be summarized in several areas:

- This paper introduces the reinforcement learning technology and agent technology in the area of self-adaptive systems research, proposes self-adaptive mechanism of self-adaptive system under uncertainty environment, and the corresponding learning algorithm for self-adaptive agent to support the learning process.

- This paper develops a specific application, presented the development process of the case, showing the actual performance of the self-adaptive agent with learning ability, verify the legitimacy and effectiveness of the methods described in this paper.

## 6. Acknowledgment

## 7. References

[1] Szepesvári C., "Algorithms for reinforcement learning", *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 4 no. 1, 2010, p. 1-103.
[2] Meignan D., Koukam A., Créput J C., "Coalition-based metaheuristic: a self-adaptive metaheuristic using reinforcement learning and mimetism", *Journal of Heuristics*, vol. 16 no. 6, 2010, p. 859-879.
[3] Mazeiar Salehie., Ladan Tahvildari., "Self-Adaptive Software: Landscape and Research Challenges", *ACM Transactions on Autonomous and Adaptive Systems*, vol. 4 no. 2, 2009, p. 1-42.
[4]Kabysh A., Golovko V., Lipnickas A., "Influence Learning for Multi-Agent Systems Based on Reinforcement Learning", *International Journal of Computing*, vol. 11 no. 1, 2012, p. 39-44.
[5]Smarsly K., Law K H., Hartmann D., "Multiagent-based collaborative framework for a self-managing structural health monitoring system", *Journal of Computing in Civil Engineering*, vol. 26 no. 1, 2011, p. 76-89.