Solutions SHaring in Multi-agent System

Hanbing Deng
College of Information Science and Engineering
Northeastern University
Shenyang, China
deng.hb@neusoft.com

Xia Zhang, Jiren Liu, Qian Zhang
State Key Laboratory of Software Architecture
Neusoft Incorporated Company
Shenyang, China

***Abstract*—In this paper, we will do some researches on the relationship between action sequence and system efficiency in multi-agent system. Usually, when an agent gets a task, it will calculate all the feasible action sequences and choose the best one whether the task has used to appeared in this system before. It wastes a lot of time on re-calculating and re-comparing. So we propose solution sharing mechanism to make all the agents share the same solution space which keeps the current optimal plan for tasks. After achieved new tasks, each agent will have only one calculation and comparison, and then choose the better current action sequence. The reduction of time and better action sequence will improve the efficiency of multi-agent system.**

Keywords-Multi-agent system (MAS), Solution sharing mechanism, Solution space, Action sequence, Current optimal solution

I. INTRODUCTION

Artificial intelligence (AI) research has gone through more than 50 years. During the 70s of the 20th century, expert system and knowledge engineering had impelled AI to achieve a prosperous period. To the real world, AI community knows that interpretation and simulation of human intelligent actions were the long-term goal of intelligent system establishment[1]. However, reviewing the existing methods and techniques of AI, we could only solve the ideals problems. In 1986, Minsky propose the concept of Agent in "Thinking Society"[2]. He said that correlative individuals in society could solve problems through consultation, and these individuals are agents which have social interaction and intelligence. Wooldridge and Jennings considered that agent should have autonomy, social interaction and responsiveness[3]. Rusell thought agent could choose correct actions through perception of environment[4].

Multi-agent systems are composed of multiple interacting computing agents. Agents are computer systems with two important capabilities. First, they are capable of autonomous action. They can decide for themselves what they need to do in order to satisfy their design objectives. Second, they are capable of interacting with other agents, not simply by exchanging data, but by engaging in analogues of the kind of social activity. In this paper, we focus on the action sequences of multi-agent system, and propose the solution sharing mechanism to prove that sharing solution space will reduce calculation and comparison time which have relationship with efficiency of multi-agent system.

II. MULTI-AGENT SYSTEM MODEL

In order to get the agent to do the task, we must communicate the desired task to the agent. This implies that the task must be specified by us in some way. An obvious question is how to tell the agent what to do. The obvious advantage of this approach is that we are left in no uncertainty about what the agent will do; it will do exactly what we told it to, and no more. But the disadvantage is that we have to think about exactly how the task will be carried out ourselves, if unforeseen circumstances arise, the agent executing the task will be unable to respond accordingly.

In order to enhance autonomy of agent, we add perception ability to the individual elements which can make them have self-perception for the environment and response to the environment with proper behavior. In the ideal case, the agent with perception ability would execute certain tasks without assists. They can make the appropriate responses when they meet some extraordinary events. But the disadvantage is that the autonomy technology is complex and the solutions space may be too large.

Basing on the perception ability, we design a solution sharing mechanism for MAS. With this mechanism, as shown in Fig.3, agents create feasible plans (action sequences) through task or environment information, and then compare them with the current plan. The solution library stores the better one which would be a manual for carrying out tasks.

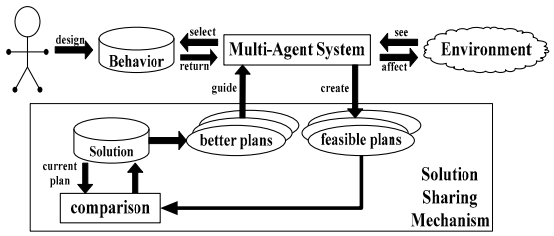

Figure 1. Solution Sharing Model of Multi-Agent System

III. SOLUTION SHARING MECHANISM

In order to specify the function of solution sharing mechanism in MAS, we will formalize the elements of MAS as follows.

© the authors

Environment may be in any of a finite set of discrete, instantaneous states. And in this paper, environment can be an infinite set E with numerable states. And E={e, e', …}.

Multi-agent system is the system which contains some individual, autonomous and interactive elements. We define that MAS should be a set of finite agents, and use MA={a ,a' ,…} to denote MAS.

Action library is a set of actions. Each agent of MAS must have the same action set. So we use set AC={c ,c' ,…} to denote action library here.

For a certain target, agents will choose action from action library first, and then composite these actions to get an ordered action sequence. We use action sequence to denote plan. For example, we assume that c0, c1, c2 and c3 are the elements of AC, we said that p: c0→c1→c2→c3 is a plan.

The basic model of agents interacting with their environments is as follows. The environment starts in some states, and the agent begins by choosing an action to perform on that state. As a result of this action, the environment can respond with a number of possible states. However, only one state will actually result, the agent does not know in advance which it will be. On the basis of this second state, the agent again chooses an action to perform. The environment responds with one of a set of possible state, the agent then choose another action, and so on. A run, r, of an agent in an environment is thus a sequence of interleaved environment states and actions: r: e0→c0→e1→c1→…→cn-1→en. From sequence r, we could get an initial state (e0), end state (en) and a plan (c0→c1→…→cn-1). And we will define solution as follows.

Solution is a set of all the feasible plans which will make MAS change the environment from initial state (ei) to terminal state (et). We use set S to denote solution and |S| to denote the amount of plans in this set. As we see, |S| is determined by both the states amount between ei and et and the average action number between two neighbor states. For example, we assume that there are n states in a run (include ei and et), and the average amount of action between neighbor states is m, and then we can get |S| = mn-1. Agent ai which belongs to MA will choose the best solution from the mn-1 plans. It will cost agents more time in comparing than running. Selecting the best action sequence implies search over the space of all possible plans. And that is why we need to propose a mechanism to optimize the selecting process.

Assuming that, MAS have a solution library which could store and share plans during MAS running. When agent ai had get a task (ei→et), it will input the initial state (ei), terminal state (et) and action set (AC) into the reasoning function, and then get a random feasible (may be not the best) plan (pi). After that, ai checks the solution library and estimates whether the library has plan p which maps task (ei→et). Agent ai compares pi with p, then gets the better plan and renew the library. If library has no target plan, ai will execute pi and then push pi into the library. The algorithm can be specified as follows.

Algorithm:
1. *agent* ← **getAgent**(*MA*); /**MA*: agent set*/
2. *action* ← **getAction**(*AC*); /*AC: action set*/
3. e_i ← **initial**(*E*); /**E*: Environment*/
4. e_t ← **terminal**(*E*);
3. *task* ← **getTask**(e_i, e_t);
6. *plan* ← **feasiblePlan**(*agent, task, action*);
7. **WHILE isPlanExist**(*lib, task*) **DO** /*lib: solution lib*/
8. *lib_plan* ← **getPlan**(*lib, task*);
8. *best_plan* ← **compare**(*plan, lib_plan*);
9. **renew**(*lib, task, best_plan*);
10. **run**(*agent, best_plan*)
11. **END WHILE**
12. **renew**(*lib*, *task, plan*)
13. **run**(*agent*, *plan*)

Annotation:

isPlanExist(*lib, t*)	return a Boolean value (true or false) to judge that if there is a plan existing in the *lib* can accomplish task *t*;
getPlan(*lib, t*)	return a plan which maps task *t* in *lib*;
compare(p_1, p_2)	return the better plan from p_1 and p_2;
renew(*lib, t, p*)	renew *lib* with plan *p* which maps task *t*;
run(*a, p*)	agent *a* run with plan *p*;

Figure 2. Solution Sharing Algorithm

According to the algorithm, the best current plan hold by the solution library will give each agent a short cut to carry out tasks. To the existed target, agent only chooses a random feasible plan to compare with the existed plan, and that reduce the checking time between two neighbor states from m to 1. Then we can get |S| = 1.

IV. EXPERIMENT AND IMPROVEMENT

The experiment scenario is a two-dimensional flat space which has some random distributed tasks and blocks. We assume that each agent could move freely in this space, and do not collide with other agents and blocks. When an agent detects some tasks, they will move to the nearest task from its current position and accomplishes the task. As shown in Figure 3, the blue ones with circular outer edge denote agents, the red ones denote tasks and the black ones denote blocks.

© the authors

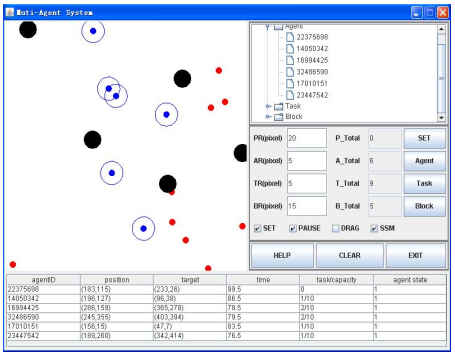

Figure 3. Multi-Agent System experiment

We set this experiment to validate whether the efficiency of system which has solution sharing mechanism (SSM-MAS) is better than the traditional one (T-MAS).Then we give the data charts to explain why SSM-MAS has advantages in agents running.

For T-MAS, when an agent detects a task, it needs to calculate the all the feasible action sequences, and then choose the best one from these plans to run. On the contrary, for SSM-MAS, when an agent detects a task, it will firstly get a feasible action sequence (may be not the best plan), and then compare this plan with the one which maps the same task in solution library. Solution space will give the better one back to agent. We record the time when all the tasks have been accomplished. In this paper, we give 10 agents to finish the work and record the time in different points of tasks amount. The time chart is shown in Fig.4(a).

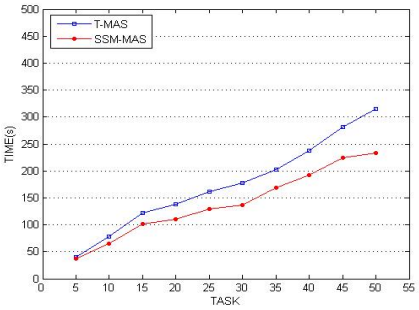

Figure 4. Time tendency chart (|MA|=10)

As we seen, within the same amount of agents, the time curve of T-MAS and SSM-MAS have rose when task amount increase. But the curve of SSMAS was smoother than T-MAS when tasks increasing. In the ideal case, the time function of T-MAS can be specified as formula (1):

$$TM = n*Tc+Tr \quad (1)$$

Tc denotes comparison time and Tr denotes running time. With the solution sharing mechanism, we can make n=1 and the time function of SSM-MAS can be specified as formula (2):

$$TS = Tc+T'r \quad (2)$$

In T-MAS, agent uses n*Tc in comparison and gets the best plan. And in SSM-MAS, agent usually could not get the best plan. So we can get $T'r \geqslant Tr$. But during the SSM-MAS running, there so many agents running in the system, similar tasks will appear frequently, the plans (action sequences) will be continually updated with better plans that make T'r get closer to Tr. Shown in Fig.5.

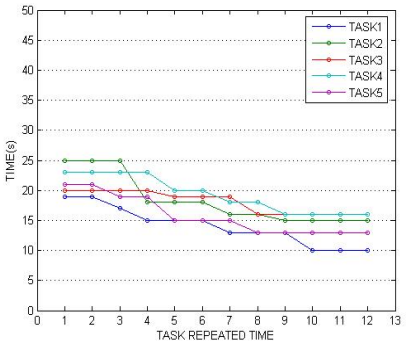

Figure 5. Time of the same tasks in SSM-MAS

SUMMARIES

Solution sharing mechanism allows action sequences composition and plan comparison will only happen one time when agent gets a certain task. Agents in the system share the same plans, and the solution library can update the records with the better plan. Better plan will guide agents to finish the job with the least time. Through the experiment and data record we can prove that, solution sharing mechanism could save the time and improve the efficiency of multi-agent system.

REFERENCES

[1] Wooldridge, M., An Introduction to Multi-Agent System, John Wiley&Sons [M], 2002, PP:1-8

[2] Minsky, The Society of Mind, Simon and Schuster [M], 1986, PP:1-12

[3] Wooldridge, M., Jennings, N. R., The cooperative problem solving process. Journal of Logic and Computation [J], 1999, PP:563-592.

[4] Russell, S., Norvig, Artificial Intelligence: a Modern Approach, Prentice-Hall [M], 1995, PP:1-10

© the authors