第36卷第2期 中国科学技术大学学报 Vol. 36, No. 2
2006年2月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Feb. 2006

Article ID: 0253-2778(2006)02-0193-05

MAS approach to traffic simulation*

ZENG Jian-qin[1,3,4], LIU Kun[2], WANG Guo-cheng[4], FANG Ting-jian[3]

(1. *Department of Automation, USTC, Hefei* 230027, *China*;
2. *The UAV Battalion, Artillery Academy of PLA, Hefei* 230031, *China*;
3. *Institute of Intelligent Machines, Chinese Academy of Science, Hefei*, 230031, *China*;
4. *The UAV Teaching and Research Center, Artillery Academy of PLA, Hefei* 230031, *China*)

Abstract: The main aspects of the multi-agent system(MAS) were agent-oriented, combined with formal program language Visual C++. A functional model of traffic system was constructed based on MAS, and on the basis of MFC library of Visual C++, the agent class that has the basic functions of cooperation, communication and autonomy is also designed.

Key words: multi-agent system; BDI(belief, desire and intention)model; traffic simulation

CLC number: U121; TP391.9 **Document code**: A

MAS 技术在交通仿真中的应用

曾建勤[1,3,4], 刘 琨[2], 王国成[4], 方廷健[3]

(1. 中国科学技术大学自动化系, 安徽合肥 230027; 2. 解放军炮兵学院无人机队, 安徽合肥 230031;
3. 中国科学院合肥智能机械研究所, 安徽合肥 230031; 4. 解放军炮兵学院无人机教研室, 安徽合肥 230031)

摘要: 运用 MAS 技术, 构建了交通系统的 MAS 功能仿真模型, 并在 Visual C++ 对象类的基础上设计了 agent 类. 仿真结显示, 这一技术能较大地提高交通仿真的精度.

关键词: MAS; BDI; 交通仿真

0 Introduction

Traffic problems have been attracting an increasing amount of intention. Many researchers are interested in using modern science and technology to relieve traffic pressure. But it is difficult to resolve traffic problems because the traffic system is a complicated, dynamical, non-linear and stochastic one.

Modeling and simulation constitute an approach which is valuable for tackling traffic problems. Many traffic simulation systems have been developed, such as UTCS-1, MISTRAN, TJTS, etc., most of which are object-oriented. Object-oriented programming is the main method used in simulation, but it can not handle intelligent objects because of its lack of autonomy and independence. In the traffic system, the driver-

* **Received**: 2004-03-15; **Revised**: 2005-05-13
Foundation item: Supported by "863" Foundation(2003AA001021).
Biography: ZENG Jian-qin, born in 1975, PhD candidate. Research field: traffic simulation. E-mail: zjq7523@sohu.com
Corresponding author: FANG Ting-jian, professor. E-mail: tjfang@iim.ac.cn

vehicle entities are typical intelligent objects that can not be simply handled as objects.

The choice of MAS technology derives basically from two observations. First, the traffic system is a complex system that involves a set of distributed and interactive entities. Secondly, agents have intelligence that can describe intelligent objects. MAS approach is a new but promising method for traffic simulation by which we can develop intelligent traffic simulation software.

1 Construction of the MAS based traffic model

The complex system can be divided into a number of distributed, interacting, intelligent and autonomous entities. Vehicles, roads, intersections, bus/taxi stops, pedestrians, traffic signals and traffic marks make up the traffic system.

MAS is now a mature technology used in the domain of distributed and intelligent control. MAS accords with the traits of a traffic system and is a proper method for constructing traffic models. With the theory of MAS, the traffic system is divided into a set of agents that communicate and coordinate with each other.

1.1 The functional structure of agents

For simplicity, we do not take pedestrians into account. The traffic system can be divided into several classes: vehicle class, road class, intersection class, bus/taxi stop class, traffic signal class and traffic mark class. Road class, intersection class, bus/taxi stop class, traffic signal class and traffic mark class only send their state information to vehicles, so they can be regarded as information agents[1,2]. Besides message-exchange, vehicle class must act on the messages after the procedure of intelligent deducing. So vehicle class can be seen as smart agents[1,2].

The construction of agent is based on the BDI (belief, desire, intention) models of rational agents[3,4]. Belief stands for the current states of the agent. Desire denotes the expected states of agent under the stimulation from the environment and other agents. Intention represents the actions that agents take according to their aim and messages obtained from the environment and is realized by knowledge/rules base.

Knowledge/rules base stores the rules and constraints that are the relationship between the messages and actions. In the traffic system, one message can trigger several actions. Sometimes, one action needs several messages as the premise. So there must be a complicated deducing procedure to ensure the correct mapping between messages and actions. In this paper, the mapping procedure is realized by TriggerMethod() that is a section of the program.

Autonomous procedure embodies the autonomy of agents that is the basic difference between agent and object. Agents have their own self-control mechanism while objects do not. Autonomy enables agents to modulate their states and actions according to messages and extend tremendously the scope of objects. So autonomous agents provide an effective means to simulate the vital objects.

Message-exchange is the basis of cooperation among agents. In the traffic system, there are two types of messages: public messages and local messages. Public messages (for example, the red/green messages sent by traffic signal agents) impact on all the vehicle agents and are exchanged via blackboard architecture (BBA), a typical model in the domain of artificial intelligence (AI). Local messages (for example, the messages of velocity and position) sent by vehicle agents only impact on their neighboring vehicle agents. For local messages, BBA is still feasible. But local messages are exchanged via message passing (MP) in order to minimize costs. BBA is an indirect method to exchange messages, but MP is a direct message-exchange method among agents.

Fig. 1 shows the functional structure of agents based on the BDI models of rational agents[3,4]. Autonomy, communication and cooperation are the basic functions of agents. Agent is the extension of

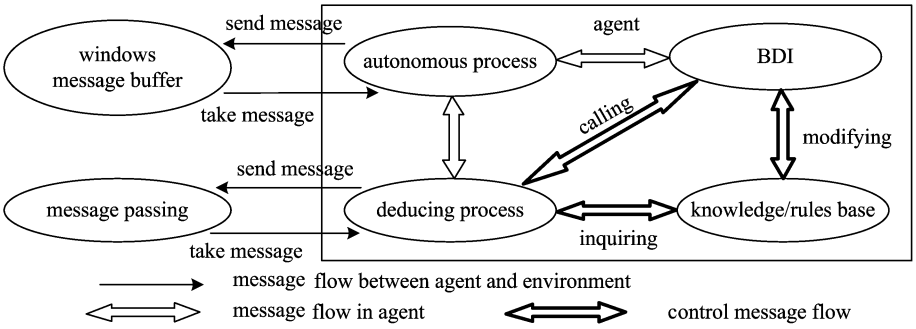

Fig. 1 Functional structure of Agent

object and can be viewed as an autonomous entity based on message-exchange and action-drive[5,6].

1.2 MAS based traffic system model

As mentioned above, the agents fall into two types: information agents and smart agents. Each agent has the traits of autonomy, communication and cooperation. Agents have their own resources and action control mechanism, so they can make decisions and take actions according to their own states and messages obtained from outside without external control.

The MAS-based traffic system model comprises many agents and has its own mechanism of communication and cooperation. When conflicts happen, these agents negotiate and coordinate with each other according to predetermined rules, criterions and ways, which is based on message-exchange. In this paper, the MAS-based traffic system model adopts distributed structure. Each agent in the model is equal and has its own resources, state information and results. So, the whole goal and the mission of an agent can be achieved based on cooperation. Fig. 2 shows the functional structure of the MAS-based traffic system model.

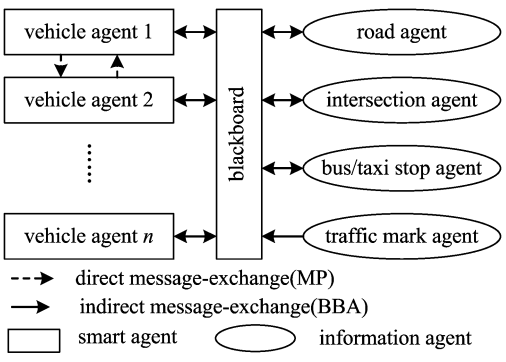

Fig. 2 Functional structure of MAS-based traffic system model

For MAS, the core technologies are communication and coordination. Communication is the basis of coordination. Information agents only communicate with smart agents via BBA. Smart agents can communicate with not only information agents indirectly but also other neighboring smart agents directly. In Fig. 2, we suppose vehicle agent 2 is next to vehicle agent 1.

2 Design of the structure of agent

Agent can be regarded as the extension of object. With the help of the mechanism of combination of classes used in object-oriented programming (OOP), different agents can be viewed as different classes and objects. We use Visual C++ 6.0 as the develop environment. Autonomy, communication and cooperation are the basic functions of agents.

2.1 Realization of autonomy

The difference between agent class and object class is that the construction function of agents must have an autonomous process. The autonomous process starts when an agent is generated. So an agent can run by itself. The autonomous process is designed as follows:

```
Virtual Void Agent Class:AutoProcess()
{BOOL HaveMessage;//define a variable to signify the
    state of GetMessage()
HWND hWnd;
do
    {HaveMessage=GetMessage (
```

```
        LPMSG lpMsg,    //content of the message
        HWND hWnd,      //handle of the window
        UINT wMsgFilterMin, //the first message
        UINT wMsgFilterMax //the last message
        );
    if(HaveMessage==true)
      TriggerMethod();
    BOOL PostThreadMessage(
      UINT message,//ID of the message defined by
        users
      WPARAM wParam,//the first parameter
      LPARAM lParam // the second parameter
      );
    }While(conditions);//use the handle of thread as
      the condition
}
```

Both GetMessage() and PostThreadMessage() are the self-contained functions of the Windows operation system. GetMessage() gets messages from the message buffer while PostThreadMessage() sends messages to the message buffer.

2.2 Realization of communication

The message buffer of the Windows operation system is used as the BBA by which agents communicate. As we know, no program can use the message buffer directly. In order to use the message buffer, two functions, GetMessage() and PostThreadMessage(), are needed. Visual C++ provides us with the two functions.

Message-drive is a typical trait of programming under the Windows operation system. A message is an integer with the length of 16 bits. Different values stand for different messages. All the messages used by the Windows operation system are defined in the file named windows.h. We also define our own messages in traffic simulation software. After calling the function of RegisterWindowMessage(), these messages have no difference with Windows messages. One message stands for only one state. The structure of messages are as follows:

```
MessageStructure
{BYTE MessageName;//name of the message
 UINT MessageNumber;//ID of the message
 HWND hwnd;//represents the window which receives
   the message
 HWND hwnd;//stands for the window which sends
   the message
 DWORD time //time at which the message was sent
 BYTE Message;//content of the message
 WPARAM wparam; //denotes the origin agent
 LPARAM lparam;//represents the objective agent
}
```

The cooperation among agents is realized by sending or receiving messages. For example, when the traffic light turns red, the traffic signal agent will send the message of red light to all vehicle agents. Then each vehicle agent will take certain actions after a deducing process.

2.3 The structure of agent

Information agents are simpler than smart agents because they have no knowledge/rules base and deducing procedure. But information agents have an autonomous process and exchange messages with BBA. Besides message-exchange, smart agents must deduce on the basis of messages and take actions. The following is the structure of vehicle agents which belong to smart agents. The structure is based on the BDI model and embodies the functions of autonomy, communication and cooperation.

```
class Vehicle:public CwinThread
{
 DECLARE_DYNCREATE(Vehicle)
 protected:
 Vehicle();//construction function
 public://Belief—the current states, such as color and
   type
 COLORREF  VehicleColor;//the color of the vehicle
   agent
 BYTE      VehicleType;//the type of the vehicle
   agent
 ……
 public://Desire—the expected states of the vehicle
   agent
 FLOAT          ExpectantAcceleration;//expected
   acceleration
 ……
 public:
 // ClassWizard generated virtual function overrides
```

```
//{{AFX_VIRTUAL(Vehicle)
public:
virtual BOOL InitInstance ( );//initialization of the
    vehicle agent
virtual int ExitInstance();
//}}AFX_VIRTUAL
protected://Intension—represents the actions that
    agent takes
FLOAT VehicleMoveA(
  FLOAT VehicleAcceleration,
  FLOAT VehicleCurrentAngle,
  BYTE  VehicleDriveCapacity
  )     //return the value of the acceleration
.....
public:
virtual ~ Vehicle ();// destruction function
DECLARE_MESSAGE_MAP() //message mapping
}
```

CwinThread is the thread class in MFC. Each object of the Class is a thread of the program. In order to use the multi-thread technology in traffic simulation software, Vehicle Class takes the CwinThread as its basic Class. The autonomous process is placed in the construction function.

3 Simulation

On the basis of the MAS-based traffic system model, we developed an intelligent traffic simulation software of an isolated intersection named ITS1.

In order to test the effect of ITS1, we developed another traffic simulation software named TS1. The only difference between ITS1 and TS1 is that TS1 is not MAS-based. The inputs of the two simulation programs are the traffic parameters of the intersection of Sanxiaokou in Hefei(the capital of Anhui Province). The two programs have the same inputs. Simulation time is one hour. The results of the two programs and the number of vehicles detected by the detectors are shown in Tab 1. Fig. 3 shows the positions of the four detectors. We can easily see that the simulation precision can be improved by 10 percent when using the MAS-based ITS1.

Tab. 1 Simulation results

position of detector	1	2	3	4
number detected by detector	1 115	1 283	829	432
results of non-intelligent simulation(TS1)	987	1 429	733	494
results of intelligent simulation(ITS1)	1 108	1 310	815	441
improving amount/%	10.9	9.26	9.89	12.3

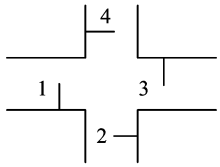

Fig. 3 The position of the detectors

4 Conclusion

In this paper, we established the MAS-based traffic system model and constructed the agent class that has the functions of autonomy, communication and cooperation. The simulation results show that the technology of MAS is suitable for traffic simulation. The introduction of MAS enables us to develop large-scale traffic network simulation software with high precision.

References

[1] Ramaswmy S, et al. A distributed agent-based simulation environment for interference detection and resolution[J]. Simulation, 2001, 76(6): 358-370.

[2] Shoham Y. Agent-oriented programming[J]. Artificial Intelligence, 1993, 60(1): 51-92.

[3] Genesereth M R, et al. Software agent[J]. Communication of the ACM, 1994, 37(7): 48-53.

[4] Chan H C, et al. Software reuse using C++ classes: the question of inheritance[J]. The Journal of System and Software, 1997, 38(3): 117-126.

[5] Bratman M E. Intentions, Plans and Practical Reason [M]. Cambridge, MA: Harvard Univ. Press, 1987.

[6] Jones P M, Mitchell C M. Human-computer cooperative problem solving: Theory, design, and evaluation of an intelligent associate system[J]. IEEE Transactions on Systems, Man and Cybernetics, 1995, 25(7): 1 039-1 053.