RoboCupRescue 2011 – Rescue Simulation League Team Description <HfutEngineRescue (P.R.China)>

<Wei Liu,Shoutao Wang,Buhui Xu,Yan Wang>
Artificial Intelligence & Data Mining Lab
Hefei University of Technology ,China
hfutliuwei@gmail.com
http://robot.hfut.edu.cn

Abstract. In this paper, we describe some features of HfutEngine, a newer in RoboCup Rescue Simulation. After enjoying some competitions and technical contacts with other teams, we have done a lot to improve. Model designing and algorithm implementation such as message encoding, path searching, lifetime prediction and partitial strategy are discussed. These ideas and some other new approaches will be shown in detail.

1. Introduction

HfutEngineRescue is developed by the Artificial Intelligence Laboratory of School of Computer & Information of Hefei University of Technology and it has participated RoboCup China Open twice. And we experience RCRSS from 0.49plus version to 1.0 version. In order to make contacts with more great teams and to improve ourselves, we take the chance enjoying the RoboCup WorldCup 2011.

2. Review of RoboCup Rescue Simulation Platform

RoboCup Rescue Simulation System (RCRSS) is a Multi-Agent System(MAS) of urban disasters, containing multi-tasking heterogeneous agents and it assigns different tasks to different agents. Moreover, the heterogeneous agents should share messages with each other to performance a better job. For example, PoliceForce should clear blockades for AmbulanceTeam and report injured civilians while AmbulanceTeam should rescue injured PoliceForce and report blockades.

RCRSS1.0 has been released in RoboCup WorldCup 2010. Compared with last version, the new version makes many changes. It considers blockade and road as area rather than simple point or line. In addition, the message channels have random noise and vision information is less.

Take above changes into consideration, our team is based on rescucore2 which is provided by RCRSS version 1.0.

3. What we have done after last competition

During 18th to 20th 2010, we participated RoboCup China Open 2010 Competition in Ordos, Inner Mongolia. Under our day and night effort, we entered final and got a good rank. After the competition, we made a conclusion and tried more to improve.

Firstly, we clarified the action logics of all agents and made them more clear and more effective. Then, we implemented the partition strategy and improved the efficiency of communication system by using bit transportation mode. The most important change was the design and implementation of deadtime prediction. Besides, we read TDPs of SEU-RedSun, ZJU-Base, MRL and other teams and found that PSO might be helpful for us.

With above work, the efficiency of our team have been improved. More details will be introduced below.

4. Agent skills

Path Searching

We use heuristic search algorithm, A* is the main method, to choose agent's path instead of simple path search algorithm. Five main procedures are listed below:

1)Take self-position as the start point.

2)Search on a map structure which is composed of roads and buildings.

3)Assign each node a value. Blockade area, road area and direction should be considered.

4)Sort nodes by value at the beginning of every cycle.

5)The direction is used for quickening the search,.If the current node is on the negative direction of target node, it will defers.

The Strategy of Extinguishing Fires

In order to control the spread of fire, we sort the firing buildings according to their connectivity properties with other buildings. The buildings that are connected directly will be considered as an area. On the basis, We find the buildings on fire at the outside edge of an area and give them the most high priority to be extinguished in case of the fire spreading to other regions. On contrary, the buildings whose adjacent areas are in safe state are in less higher priority. Of course, we take the population in fire areas and adjacent areas into serious consideration and change their priorities.

Deadtime Prediction Based on PSO

How to Use PSO?

Accurate deadtime prediction is very important to AmbunlaceTeam's work and the whole efficiency of a team. After comparing some prediction algorithms , we finally use the particle swarm optimization (PSO). We do this as follow:

1)Initialize some particles which are triple(H, D, B) where H is the hp property of a civilian, D is the damage property and B is the buridness property.

2)Update particles by applying the model and formulas as follow:

$$(H, D, B)_{n+1} = (H, D, B)_n \tag{1}$$

$$D_{n+1} = w \times D_n + r_1 \times c_1 \times (pbest - H_n) + r_2 \times c_2 \times (gbest - H_n) \tag{2}$$

$$B_{n+1} = w \times B_n + r_1 \times c_1 \times (pbest - H_n) + r_2 \times c_2 \times (gbest - H_n) \tag{3}$$

$$H_{n+1} = H_n - a \times D_n \times D_n - b \times B_n \times B_n \tag{4}$$

where n is the particle generation, w is inertia factor and it is between 0.4 and 0.9 usually, r_1 and r_2 are two random double numbers where $r_i \in [0,1] (i = 1,2)$, c_1 is the self-awareness ability that describes the influence of particle's best position, c_2 is the social-awareness ability that describes the influence of social best position, $pbest$ is particle's best position, $gbest$ is the social best position, a and b are coefficients of the model.

3)Check H, D and B every cycle. If some of them are illegal, remove the particles from the group and then use legal particles to replace them. Here, we make a rule: $|H_{n+1} - H_n| < 300$, $|D_{n+1} - D_n| < 10$ and $|B_{n+1} - B_n| < 50$.

4)If all particles are illegal, reinitialize the particles and begin a new cycle.

The procedure of this algorithm is shown in **Fig.1.**

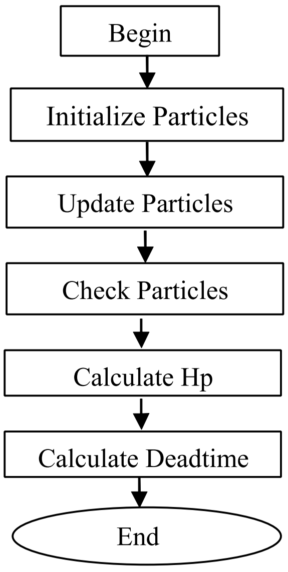

Fig.1. The Procedure of PSO

How to Initialize Particles?

Generally, there exist two methods to initialize particles and set the initial values. One is from observation and another is from off-line table. Here, we use the latter method. Firstly, we make three tables of hp, damage and buridness property. The values in each table are from experience and of a large range. Then we can get a particle's initial values from the three tables based on some rules.

How to Get Good Coefficients?

To get a quite better result, we need a good a and b . Here, we use off-line learning to train the model to get relatively good coefficients and then use on-line learning method to modify them.

Partition Strategy

In order to improve the efficiency of PoliceForce, we implement the partition strategy. The strategy is decripted as follows:

1)Get central point of every map using the following formulas.

$$x_o = \frac{x_{\max} - x_{\min}}{2} \tag{5}$$

$$y_o = \frac{y_{\max} - y_{\min}}{2} \tag{6}$$

where (x_o, y_o) is the central point of a map, $x_{\max}$ and $x_{\min}$ is the maximum and minimum of x coordinate , $y_{\max}$ and $y_{\min}$ is the maximun and minimum of y coordinate.

2)Divide the whole map into four parts as **Fig.2** shows.

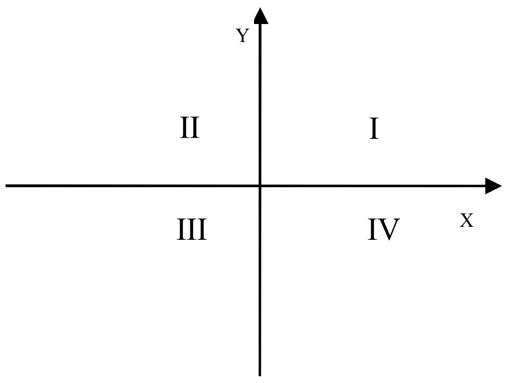

Fig.2. Partition Quadrant

3)Judge which quadrant an agent belongs to according to its coordinate.

4)Assign different work to different agents. As a result, an agent does jobs in its own quadrant first and it doesn't need to go to a far place.

5. World Model

World model plays an important role in rescue procedure, it affects the competition result directly. In our world model, there exist four parts:

1) Vision information: the perceptive messages which agents get from KA_SENSE.
2) Shared information: information extracted from messages other agents share.
3) Prediction information: predictive properties based on current properties.
4) Specific information: information the agent use to deal with its properties. For example, the minimum spanning tree which is calculated on the base of roads structure.

6. Agent coordination and communication

Communication Mode

In our communication mode, communication system is only used for sharing information without sending commands. Two kinds of information is sent in our codes: perceptive information and state information. Each agent shares perceptive information with other agents through communication system when it gets KA_SENSE messages. If an agent receives perceptive information from other agents, it uses the information like KA_SENSE messages to update its own world model. State information is composed of current location, HP, current command, etc.

Rational Control Mode

Currently, lots of teams use centralize-control mode. However, agents may lose self-decision ability if they only perform the jobs assigned by centers. Instead of centralize-control,We apply a new control mode with name of rational competition.

In this mode, agents make decisions by the following steps :

1)Sort tasks by significance based on world model. Moreover, each task has an unique calculating model.

2)Calculate the time spent on a task that an agent finish it alone, and then calculate the time the homogeneous agent spend.

3)Assign each task to an agent which can finish it most savingly. Assigned agents and tasks will never be reassigned.

4)When agents gets self-task, end decison.

If tasks are more than agents that can be assigned, we ignore the excess tasks. Otherwise, the excess agents execute assistant tasks. The assistant tasks are helping homogeneous agents to finish complex tasks and help heterogeneous agents to finish searching task.

Message Processing and Encoding

Because we don't depend on centralize-control, a good message processing and encoding mechanism is very important to us.

In our codes, at the begining of each cycle, messages sent by the agent will be saved rather than be sent directly. And agent will send the messages at the end of each cycle, so we pack messages up and wait for a fit time to send them. But how can we pack them up? As above said, the center agents are not used to control group agents now, so we use them do the packing job. After packing messages up, they re-send the package to group agents. During packing messages up, the centers must remove outdated messages.

In order to improve the utilization of channels, we encode messages bit by bit. As you know, there exist many types of agents and messages in RCRSS, so we must do something special to distinguish them. Here, we do the encoding work according to format as **Tab.1**, **Tab.2**, **Tab.3** and **Tab.4** show:

Tab.1. Agent State Message Format

MSG_HEADER(5b)	Agent ID(20b)	Location(20b)

Tab.2. Building State Message Format

MSG_HEADER(5b)	Building ID(20b)	Fireness(20b)

Tab.3. Civilian State Message Format

MSG_HEADER(5b)	CV ID(20 b)	CVHP(20bit)	CVDam(20b)	CVBur(20b)

Tab.4. Blockade State Message Format

MSG_HEADER(5b)	Blockade ID(20b)	Position(20b)

MSG_HEADER is defined as the **Tab.5**.

Tab.5. MSG_HEADER Definition

NULL_HEADER	00001
AT_HEADER	00010
FB_HEADER	00011
PF_HEADER	00100
BUILDING_HEADER	00101
CIBILIAN_HEADER	00110
BLOCKADE_HEADER	00111

7. Software Architecture

Our code is based on rescuecore2 provided by RCRSS-1.0, and thanks to the help of SEU-RedSun and YabAPI, who use ActionCommandExecption to control simulation cycle. We add skill layer between decision and action layer as **Fig.3** shows. Decision layer only generate skills but don't consider detailed action commands, the skill layer will transfer action commands according to current world model. Each action command will generate ActionCommandExecption after it's sended, the execption will end the simulation cycle, and make sure that each cycle will only send one action command, skill layer can know the action command in last cycle. Decision, skill and action layer can directly access to world modle.

8. Result

With above methods and strategies, HfutEngine is better than before. **Fig.4** and **Fig.5** show the different results between old HfutEngine and the newest version. From the two figures, we can find the improvement of HfutEngine. Although we are not better enough than some famous teams, we will continue to make improvement and try our best to catch up with them.

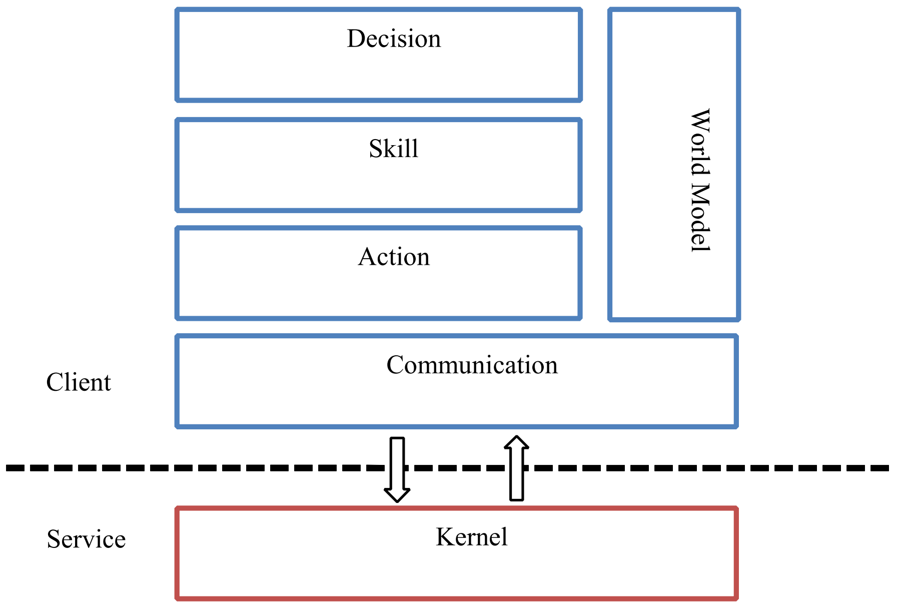

Fig.3. Software Architecture

9. Acknowledgements

Thank all of the server developers for maintaining the rescue simulation server and for their kindly technical supports. Besides, we show our gratitude to SEU-RedSun for their TDP and source codes.

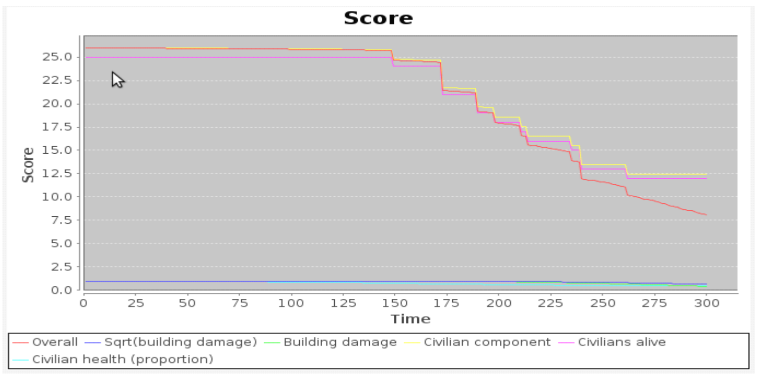

Fig.4. score chart of old HfutEngine

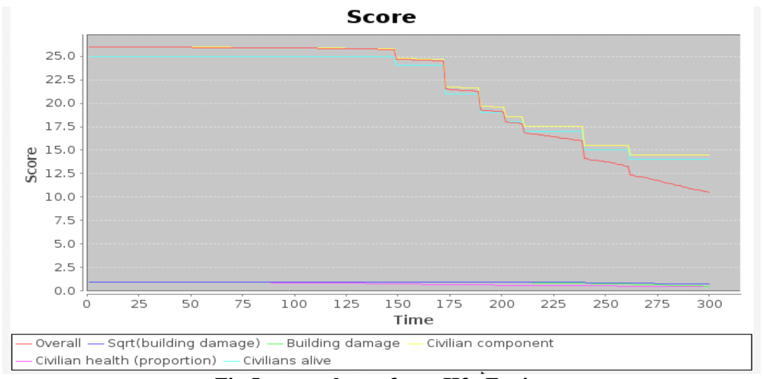

Fig.5. score chart of new HfutEngine

10. References

[1] The RoboCup Resuce Technical Committee.RoboCup Rescue Simulator Manual version 0 reversion 4,4-59(2000)
[2] Takeshi Morimoto.How to Develop a RoboCupRescue Agent for RoboCupRescue Simulation System version 0 1st edition,1-9(2001)
[3] Jun Peng,Fu Jiang,Ya Liu,Xiaoyong Zhang,et al.RoboCupRescue 2009 - Rescue Simulation League Team Description CSU Yunlu(China),1-7(2009)
[4] Daqi Guan,Qingfa Meng.RSL 2009 – Rescue Simulation League Team Description <SEU_RedSun (China)> ,1-10(2009)