Center Based Task Allocation in Multi Agent Systems

M.E.Shiri, A.Behzadian Nejad, H.Houshyarifar, E.Pasbani,
Department of Computer Science
Faculty of Mathematics & Computer Science
Amirkabir Univercity of Technology (Tehran Polytechnic),Tehran , Iran
http://www.abn.50g.com/rayan
{shiri, houshyarifar}@aut.ac.ir
{ali_behzadian, ebrahim_61}@yahoo.com

Abstract. There are some problems in RoboCupRescue Simulation system such as task allocation, path planning, inter-agent communication and so on. We use some algorithms to solve these problems. This paper describes the design and implementation of Rayan RoboCup Rescue Simulation team and the algorithms used to solve problems relevant to this system.

1 Introduction

RoboCupRescue simulation is a multi-agent system involves heterogeneous agents that work together as a team to reduce disaster of an earthquake in the city.Agents in this system are police force, police office, ambulance team, ambulance center, fire brigade and fire station. With 6 different types of agents and almost more than 30 agents, communication between agents and task allocation are difficult problems. To extinguish a fire, save a civilian's life or clear a road, agents must go to the location of fiery building, civilian or blocked road, therefore path planning is an important skill of all platoon agents. There are some traditional algorithms that give us all pair shortest paths (like Floyd). But these algorithms are inefficient and they need whole graph information at the start.

This paper is organized as follows: The overview of cooperation and task allocation system is presented in section 2. In section 3 we describe inter-agent communication language used in the system. In section 4, the algorithms of path planning is described. And finally in section 5 a brief of the future work is described.

2 Center Based Task Allocation

Cooperation and task allocation are main parts of designing agents. For a good and efficient design, all agents must cooperate with each others; allocate tasks between themselves and complete work as a team.

In Rayan team we implement a task allocation system named "Center Based task allocation", in order to abandoning process of *finding tasks* and *assigning tasks* to the headquarters of platoon agents.

In this system agents either have some pre-ordered tasks that they should do it lonely. Whole the system is inter-acting with kernel and environment.

This means that all agents have real-time information about events happening in the environment around them.The scenario of task allocation divides into two parts; which are center scenario and agent scenario.

2.1 Agent Scenario

In our system agents have three states, and center assigns tasks to agents depend on their states. These states are:

2.1.1 Not busy

At this state, agent either has early completed its task or it hasn't any task to do. At this state agent will makes decision on its own information and selects tasks to do and if distinguishes that there is no urgent work at this time, it moves in the world and receives the environment's information and sending them to center (for updating center's world model) At the end of each cycle.

2.1.2 toward target

At this state if agent receives a message from center will compare it with its current work and will do which has more priority. At the end of each cycle the agent will inform the center if it has completed the assigned job or not?

2.1.3 Busy

At this state agent continues its work and inform the center that it can't accept the new task.

From the agent's point of view, so the process of each cycle is divided to

- Checking its state,
- Receiving tasks assigned by center,
- Making decision about its tasks,
- Doing task with highest priority.

2.2 Center Scenario

At the start of each cycle, center process all information received from platoon agents. Extract useful information from these data and updates its knowledge-base.In the next stage, center finds the tasks and assigns a priority to each task, and then selects agents that must accomplish these tasks.
Form the center's point of view, process of each cycle divides to:

- Processing the received data,
- Extracting useful information from these data,
- Finding the tasks and assigning priority to each task,

- Selecting the agents that must do these tasks based on factors like location and state of the agent,
- Sending commands to selected agents.

Figure 1 shows center scenario described above.

Some advantages of this task allocation system are that decision making process becomes more precisely and it's very similar to the process happens in the real world.

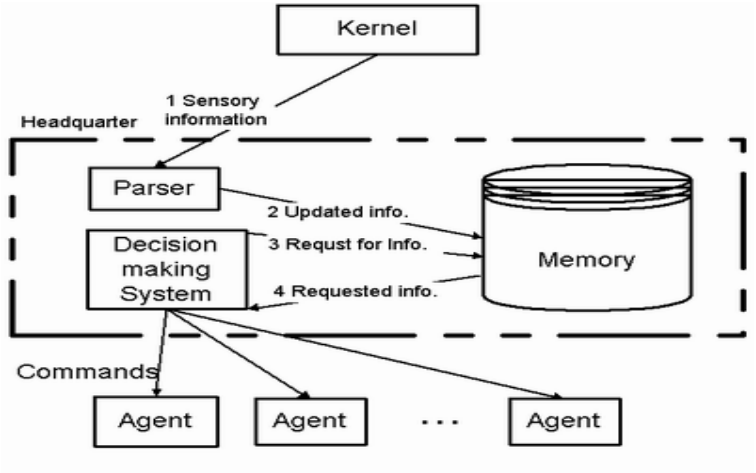

Figure 1. shows the process of center in a cycle

The disadvantage of the system is the heaviness of communications used in the system. To solve this problem agents must first avoids sending blind messages (like messages that it didn't know that they are correct or not).

And use Secondly a powerful inter-agent communication language that guarantees the efficiency of messages (ratio of contents per length and number of message).

3 Inter-agent Communication

As mentioned above, an efficient inter-agent communication needs a powerful language that supports all types of messages used between agents. For this purpose we use a language proposed by Itsuki Noda and others in [2]. In this language the format of messages is:

(*SpeechAct* : sender Agent
: receiver Agent

: content Content
...)

Some variations of ***SpeechAct*** are `inform` (that tells the sender believes content is true), `query-ref` (that tells sender asks the receiver content is true or not) and so on. For more for more details see [2].

4 Path Planning

The path planning problem is relevant to all applications in which a mobile robot should autonomously navigate. Finding the shortest path in an environment that is only partially known and changing is a difficult problem. There are some traditional algorithms that give us shortest paths in the graph such as Dijkstra's algorithm or Floyd's algorithm, but these algorithms are. In fact we can't use these algorithms on the partially known and changing graphs like maps because the cost of edges are changing during time.For solving this problem we use a method called *"Radar Path Planning"* [3].

5 Future Works

We are working on a new flexible, extensible and documented framework based on Java and we try to finish it before the competitions and if this happens we will release it on the internet to simplify developing RoboCupRescue agents.

Acknowledgements

We are thankful of members of NITRescue for their published source code of 2001.

References

[1] The RoboCupRescue Technical Committee: *Robocop-Rescue Simulator Manual*, Version 0 Revision 4.

[2] Noda, Itsuki and Takahashi Tomichi and Morita Shuji and Koto Tetsuhiko and Tadokoro Satoshi: *Language Design for Rescue Agents*.

[3] Ulrich Roth, Mark Walker, Arne Hilmann, and Heinrich Klar :*Dynamic Path Planning with Spiking Neural Networks*,. TU-Berlin, Institute fur Mikroelektronik, Jebensstr. 1, D-10623 Berlin

[4] Takeshi Morimoto: *How to Develop RoboCupRescue Agent for RoboCupRescue Simulation System* 1st edition.

[5] P. Norvig and S. Russell, "Artificial Intelligence: *A Modern Approach*, Prentice Hall, 1995.