The RAYAN Team Description For RoboCup 2003

www.rayan.behzadian.com
rayan@behzadian.com, rayan_rescue@yahoo.com

Seyyed Mahdi Tashakori Hashemi[1], Mohammad Ebrahim Shiri[2], Ali Behzadian Nejad[3], Hamid Hooshyarifar[4], Ebrahim Pasbani[5], Adrin Jalali Khoshahr[6], Mohammadreza Jooyandeh Roodsari[7], Behnam Esfahbod[8]

Department of Computer Science,
Faculty of Computer Science and Mathematics,
Amirkabir University of Technology, Tehran, Iran
www.aut.ac.ir

This project is partially supported by **NAJA Research and Development Institute**

[1]tashakori@aut.ac.ir, [2]shiri@aut.ac.ir, [3]ali@behzadian.com, [3]ali_behzadian@yahoo.com, [4]houshyarifar@aut.ac.ir, [5]ebrahim_61@yahoo.com, [6]adrin_jalali@aut.ac.ir, [7]mr_jooyandeh@aut.ac.ir, [8]behnam@esfahbd.info

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, injured 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 in a graph (like Floyd and Dijkstra). But these algorithms are very slow and inefficient.

Rayan RoboCupRescue Simulation Team is the result of a RAYAN Robotic Research Group in RoboCup field in the RoboticLab in Amirkabir University of Technology (AUT). We attempt to find best strategies to solve problems described above specially task allocation and cooperation between agents.

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 are described. Other sections are Acknowledgements and References.

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 other, allocate tasks between themselves and completely work as a team.

In Rayan team we implement a task allocation system named "Center Based task allocation", because of our main idea is based on 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.

We use one memory for our main process on the center and make real-time updates on it. It is so necessary to makes this memory updated because of our decision making system work with this memory's information. Our scenario for task allocation divides into two scenarios; these are ***Center Scenario*** and ***Agent Scenario***.

2.1 Agent's Scenario

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

2.1.1 Not Busy

In this state, agent hasn't any work and no tasks assigned to it or it has finished its work. 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 gets the environment's information and send them to center (this is for updating world model of center). At the end of each cycle, agent will sends its gathered information and its state to the center to update center memory.

2.1.2 Toward Target

At this state if agent receives a message from center, It will compare it with its current work is doing and will do one of them that has more priority and send the result to center to know that the work assigns to agent had chosen to complete or not. At the end of cycle agent sends the result to center that assigned job has completed or not and if not, why agent didn't accomplish task?

2.1.3 Busy

At this state agent without any care of messages received from center continues its work and send a message to center that it can't do the assigned task. From the agent's point of view, process of each cycle divides 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 in the previous cycle, extract useful information from these data and updates its memory. For extract true information of received information, center uses some ideas of data mining. In the next stage, center finds the tasks and gives a priority to each task, and then selects suitable agents that can accomplish these tasks and assigns tasks to these agents. Form the center's point of view; process of each cycle divides to:

- ♦ Processing the received data in the previous cycle,
- ♦ Extracting useful information from these data with data mining methods,
- ♦ Finding the tasks and giving priority to each task,
- ♦ Selecting the suitable agents for doing these tasks based on factors like location and state of the agent,
- ♦ And at the end of cycle, sending commands to selected agents.

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.

But this system has a disadvantage and it is heaviness of communications used in this system. To solve this problem -at first- agents must avoids sending blind messages (like messages that agent didn't know that their content is true or not). Second solution is using 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 format of messages is:

(***SpeechAct*** **:** **sender** Agent
: receive r 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 information about this language please see the [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 very slow and need to a large memory. In the interim, we can't run these algorithms on the partially known and changing graphs like maps because of cost of edges (roads) are changing during time.

For solving this problem we use a method called *"Radar Path Planning"* [3]. "The radar path planner is based on a simple idea: A starting point *S* and a target point *T* are given in the workspace Y. In regular intervals the start point *p* sends out a wave front traversing Y with constant speed. The wave front cannot move through obstacles. The neighbor point *P* of *S* that is first reached by the wave front must be part of an optimal path to *T*. This is because the wave front spreads to all directions with equal speed, so if *P* is reached first, it must lie on a shorter path to the origin of the wave front than any other point reached later. So *P* is added to the path W and it becomes the new starting point for the path search. If now *Q* is the neighbor point of *P* that is first reached by the wave front, *Q* is added to W and becomes the new starting point, etc. So, the path is increased by one point with each wave front until *T* is reached. Thus, an optimal path connecting *S* and *T* is built."

5 Acknowledgements

We are thankful of members of NITRescue for their published source code of 2001, Takeshi Morimoto for YabAPI, S.O.S team members, Arash Rahimi, Mohsen Izadi, … and our other close friends in the RoboticLab especially Abdolaziz Gholami for their valuable helps in algorithmic problems.

6 References

1. The RoboCupRescue Technical Committee: RoboCup-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, Institut für Mikroelektronik, Jebensstr. 1, D-10623 Berlin
4. Takeshi Morimoto: How to Develop RoboCupRescue Agent for RoboCupRescue Simulation System 1st edition.
5. Amund Tveit: A survey of Agent-Oriented Software Engineering, Norwegian University of Technology, May 8, 2001
6. P. Norvig and S. Russel: Artificial Intelligence: A Modern Approach. Prentice Hall, 1995.
7. R. Beale and T. Jackson: Neural Computing: An Introduction, Institute of Physics Publishing , 1998.
8. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Cliford Stein: Introduction to Algorithms, Second Edition, The MIT Press.
9. Hiroaki Kitano, Satoshi Tadokoro, Itsuki Noda, Hitochi Matsubara,Tomoichi Takahashi, Atsuhi Shinjou, Susumu Shimada: RoboCup Rescue: Search and Rescue in Large-Scale Disasters as a Domain for Autonomous Agents Research.