Implementation of a communication library among heterogeneous agents: NAITO-Rescue 2013(Japan)

Dai Obashi[1], Toshiyuki Hayashi[2], Nobuhiro Ito[1], Kazunori Iwata[3]

[1] Department of Information Science, Aichi Institute of Technology, Aichi, Japan.
[2] Department of Electrical and Computer Engineering, Nagoya Institute of Technology, Aichi, Japan.
[3] Department of Business Administration, Aichi University, Aichi, Japan.
chocolatelynx@gmail.com, haya815t@gmail.com, n-ito@aitech.ac.jp, kazunori@aichi-u.ac.jp

Abstract. The RoboCup Rescue Simulation (RCRS) is a multi-agent environment that simulates postdisaster relief. The agent competition of the RoboCup Rescue Simulation league is a place to evaluate various multi-agent techniques.
In this paper, we introduce two solutions for a disaster-relief multi-agent team for RoboCup 2013. First, we introduce a communication library among heterogeneous agents, in order to improve the reusability of source code for multi-agent teams. The library provides an information sharing protocol among the heterogeneous agents and, through our experiments, we can confirm that these agents could communicate with each other. Next, we introduce the structure of the multi-agent team that would use our library. Our team has adopted the x-means method for each agent to decide his own area, thereby avoiding overlapping with another agent for a distinct area. We then show the results of these experiments.

1 Introduction

The RoboCup Rescue Simulation (RCRS) is a multi-agent environment that simulates postdisaster relief [1]. The aim of the RCRS is to find effective strategies for dealing with disaster situations by developing artificial intelligence agents, and to expand the progress of artificial intelligence and robotics using new technologies introduced during the development of these agents.

The RCRS is an environment to reproduce real world disaster scenarios with agents to simulate rescue team operations. The environment is composed of disaster information such as fires and blockades, and geographical information such as buildings and roads. Agents consist of modeled rescue teams so the fire brigade agent must extinguish fires, the police force agent must clear road blockages and the ambulance team agent must rescue victims. The fire station, police office, and ambulance center must manage the relevant fire, police, and ambulance teams. To find an effective strategy, many researchers and developers have been working toward the development of multi-agent teams.

The agent competition of the RoboCup Rescue Simulation league is a place to evaluate various multi-agent techniques. In this paper, we introduce two solutions, namely a communication library and an approach for the design of our agents for a disaster-relief, multi-agent team for RoboCup 2013. In Chapter 2, we introduce the Communication Library. The purpose of this library is to improve the reusability of source code for multi-agent teams. The library provides an information sharing protocol among the heterogeneous agents and we can confirm through our experiments that these agents that used our library could communicate with each other. In Chapter 3, we describe an agent developed using our library. Our team adopted the x-means method for each agent to decide his own area. This method can avoid overlapping of agents for a distinct area and this is confirmed by experiment. In Chapter 4, we present our conclusions.

2 Communication Library

2.1 Communication problem in the RCRS

The current RCRS has several problems, including low reusability of agents. Researchers and developers have to develop an entire multi-agent team to apply their effective strategies to the RCRS. Therefore, it is important to reuse the agents of the RCRS in order to simplify the development of multi-agent teams. However, reusability is low because developers individually design unique communication standards for their own agents. To solve this problem, it is necessary to unify communication standards.

2.2 Overview of Communication Library

In order to solve the problems in the RCRS, a prototype of the RoboCup Rescue Simulation Communication System (RCRSCS) library has been developed [2]. The RCRSCS library provides a protocol for designing and implementing actions for agents and their teams. When a multi-agent team is designed with the communication library, we can prepare a center agent, which is a leader agent, and other agents, which are member agents. The center (leader) agent can give global instructions to member agents. These members can behave autonomously according to the instructions from the center agent. As a result, the RCRSCS library achieves an improvement in reusability of the multi-agent teams. The RCRSCS library is structured on the basis of the concept of FIPA ACL [3], one of the multi-agent communication languages, and this structure is shown in Figure 1.

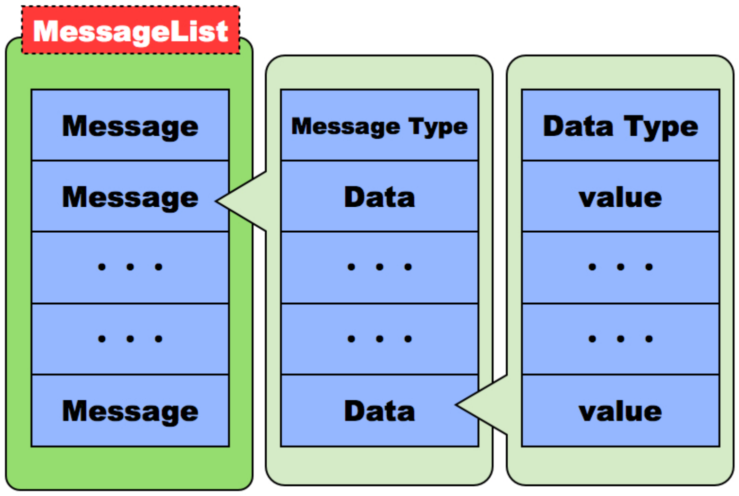

Fig. 1. Structure of RCRSCS library

Communications consist of multiple messages. Each message consists of the **Message Type** and multiple **data**. Each data consists of **Data Type** and multiple **value**s.

The **Message Type** is classified into one of three; the **Information Message** is used for sending information obtained from the disaster area to other agents and center agents, the **Task Message** is used for task (instruction) messages from a center agent to other agents while the **Report Message** is used for reporting the result of the given task back to the center agent. Each message is divided into detailed information about buildings, blockades, etc.

When a message is sent, it is automatically converted into a bit sequence so a message passing through the library does not waste resources, although message length per simulation step is restricted by the RCRS rules [4].

2.3 A problem of the RCRSCS library

The prototype RCRSCS library contains some problems, as follows:

1. Types of **Task Message**s are insufficient.
2. In message passing, some settings for multicast are required.
3. There is little documentation.
4. There are still some bugs.

In this paper, we focus on point 1. The prototype library assumes the following two points:

– There are center agents.
– There is damage caused by the disaster, which agents should overcome.

In terms of the first point, a center (leader) agent should be elected in the case where none exist. For the second point, we must assume a situation where all damages caused by the disaster have been recovered. To solve these problems, it is necessary to design new **Task Message**s.

2.4 New Task Messages

We designed new **Task Message**s to solve the problems described above and these are shown in Table 1.

The **Scout Area Message** is an order for agents to search for damage and civilians in a specified area. The **Decide Leader Task Message** is an order to elect a center (leader) agent if one is required.

Table 1. New Task Message

Task Message	**Elements**
Scout Area Task Message	agent ID, scout work area ID list
Decide Leader Task Message	agent ID

2.5 Evaluations for the library

We confirmed the effectiveness of our improved library through experiments. We asked a few teams to implement a multi-agent team using our library. We used the agents as samples and created mixed agents by combining them. Then we performed ten simulations for each agent. To evaluate our library, we compared the sample agent with the mixed agent for the ratio of the number of **Task** and **Report Message**s. If the mixed agents could correctly communicate with each other, the number of **Report Message**s became equal to the number of **Task Message**s, because one **Task Message** requires one **Report Message**. The sample agent had the ability to perform the contents of received **Task Messages**.

The results of the experiments are shown in Table 2,

Table 2. The average ratio of the report and task messages

Team	Report Message / Task Message (%)
Sample Agent	73.53
Mixed Agent	74.49

which shows the total number of **Report** and **Task Message**s in 300 steps of simulation. From Table 2, we can confirm that the result of the sample agent is close to that of the mixed agent. The ratio of **Report** and **Task Message**s is not 100% because an agent can not receive the **Report Message** if the agent died during the message passing, nor can the agent receive the message when the simulation is finished.

As a result, we can confirm that our library performed properly.

We implemented our multi-agent team as a sample agent for our library. In the following section, we describe an approach for our agent.

3 Agent

3.1 Common approach to every agent

The agent's priority is to search for damage (fire, victim, blockade) caused by the disaster. Each agent is responsible for a specific area so as to avoid searching in the same place, agents use the x-means method to decide their own area. Inputs of the x-means are the coordinates of buildings. X-means [5] is a clustering technique, which minimizes BSI (Bayesian information criterion). As a result, the buildings are divided into clusters according to the distance between buildings. Each cluster consists of buildings that are closer to its neighbors than the average distance of each building. Agents used our communication library.

3.2 Fire Brigade

Fire Brigades extinguish small fires in a specific order. First, the Fire Brigade selects small fires that they can extinguish. Second, they select fires in their own area and finally, other fires. The x-means clustering ensures that the distance between the buildings of one area and another is sufficient to contain the slow spread of fires.

3.3 Ambulance Team

The highest priority is to rescue civilians; therefore, the Ambulance Team has to rescue as many as possible. To decide which civilian the Ambulance Team has to rescue, we introduce the following approach.
(1) = (The number of steps required to rescue a civilian) +
(The number of steps required to get to the nearest refuge)
(2) = (The number of steps that the civilian can survive through)
If (1) < (2), the Ambulance Team rescues the civilian on the basis of the abovementioned two factors, in ascending order.

3.4 Police Force

First, the Police Force clears blockades that agents are buried under. Second, they clear blockades on the main roads, which are decided by the Min-Cut method [6].Third, they clear blockades that prevent agents from passing through. Fourth, they clear other blockades. Inputs of the Min-Cut are the vertices and edges of a graph. The coordinates of the intersection of roads are regarded as vertices of a graph, while connections between the roads are regarded as edges. The weight of all the edges is one. The Min-Cut is a method to find the minimum cut of an undirected, edge-weighted graph. The minimum cut is a set of edges that divide into two or more connected graph components. In other words, if the roads (edges) of minimum cut become unavailable, the agent can not get to their destination. Therefore, the minimum cut is regarded as the main roads.

3.5 Result

By clustering buildings, agents are placed evenly and properly on the map. Therefore they can search for damages caused by the disaster more quickly.

Blockades don't prevent agents from passing through so much, because the Police Force clears blockades on the main roads based on a priority basis. But the main road becomes congested, while agents can't search for damages, except for damages around main roads. We'll improve planning path problem in near future.

4 Conclusion

In this paper, we have introduced two solutions for a disaster-relief multi-agent team of RoboCup 2013. Because we have improved the expressive ability of the RCRSCS library, the problems of the prototype library are solved. We have confirmed the effectiveness of our improved library through experiments. In the future, we will confirm the effectiveness by using various algorithms.

References

1. RoboCup Rescue Simulation League. http://roborescue.sourceforge.net/.
2. Takefumi Ohta, Fujio Toriumi, "RoboCupRescue2011-Rescue Simulation League Team Description <SUNTORI(Japan)>", 2011.
3. Michael Wooldridge, "An Introduction to MultiAgent Systems", pp.140-146, Wiley, 2009.
4. RoboCup Rescue Simulation League Agent Competition 2010. Rules and Setup. http://roborescue.sourceforge.net/2010/rules.pdf, p.4, 2010.
5. Dan Pelleg, Andrew W. Moore, "X-means: Extending K-means with Efficient Estimation of the Number of Clusters", Proceedings of the Seventeenth International Conference on Machine Learning, pp.727-734, 2000.
6. Mechthild Stoer, Frank Wagner, "A simple min-cut algorithm", Journal of the ACM, vol.44, pp.585-591, 1997.