RoboCupRescue2010-Rescue Simulation League Team Description

suntori(Japan)

Takefumi Ohta, Fujio Toriumi

Graduate School of Information Science,
Nagoya University
{takefumi, tori}kishii.ss.is.nagoya-u.ac.jp
http://www.kishii.ss.is.nagoya-u.ac.jp

Abstract. The aim of RoboCup Rescue Simulation is to find effective strategies for responding to disaster situations. The system provides a platform for attempting ideas while using multi-agent systems. However, it requires high-level programming skills to develop agents for RCR Simulation due to the complexity of the task. In this paper, we describe our design of the RescueBit System, which uses the Task-Job System proposed in our previous work. This effort is motivated by the desire to increase the participation of novice competitors. Moreover,we propose the concept of FireSite, which helps agents to extinguish fires more effectively.

1 Introduction

The Robocup Rescue Simulation System (RCRSS) is one of the most complicated multi-agent systems, posing an interesting challenge for study. However, it requires a high level of programming skills and knowledge of detailed rules. Therefore, in spite of RCR's use in a competitive setting, novices are often excluded because of the difficulties involved in its development. This is one of the reasons why it has been difficult to increase the number of new competitors.

The Suntori team has designed the RescueBit System, which allows competitors to develop agents using a simple GUI method. This system is proposed for the express purpose of increasing the number of new entrants in RoboCup. By using this system, users can easily join and thus experience RoboCup Rescue.

Moreover, we designed the FireSite approach to improve the performance of the fire brigade agents. FireSite allows agents to predict the continuing spread of fires.

First, we review of the Task-Job System and summarize the RescueBit System. Next, we describe FireSite, which has improved the quality of our team's performance. Finally, we show the efficacy of our approach.

2 Scientific Contributions

2.1 Past achievements

The Robocup Rescue Simulation System (RCRSS) requires a high level of programming skills and knowledge of detailed rules. Due to these challenges, there are many difficulties in managing agents, such as the tasks of moving agents to destinations by the shortest path from scratch and extinguishing a blaze at a particular building.

The objective of RoboCup Rescue is to find effective rescue strategies under disaster conditions. However, those who develop agents have to be able to handle them freely. This difficulty effectively shuts out new participants and beginners.

Therefore, we proposed the Task-Job System, which only required developers to select the operation that agents need to perform as a set of operations, called a task. Each task represents the activity of an agent with such purposes as "Extinguish fire at building A" or "Rescue civilian B," and it is a collection of the smallest elements of an agent's action (called Jobs). As an example, let's look at the task "Extinguish fire at building A." The Process flow of the Task-Job System is shown in **Fig.1**.

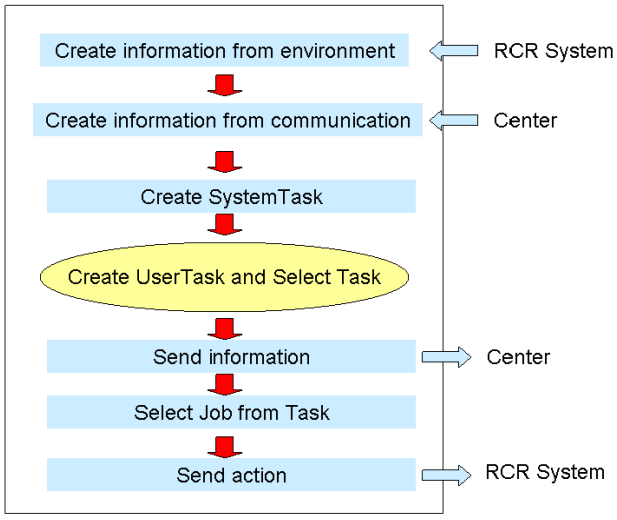

Fig. 1. Process flow of Task-Job System

Flexibility is one of the most important feature of this system, since it is needed to allow users to create original tasks. We assume that beginners develop agent teams by using given tasks, mid-level users create tasks from certain components called jobs, and experts design their own tasks from scratch for more complex behaviors (**Fig.2**)

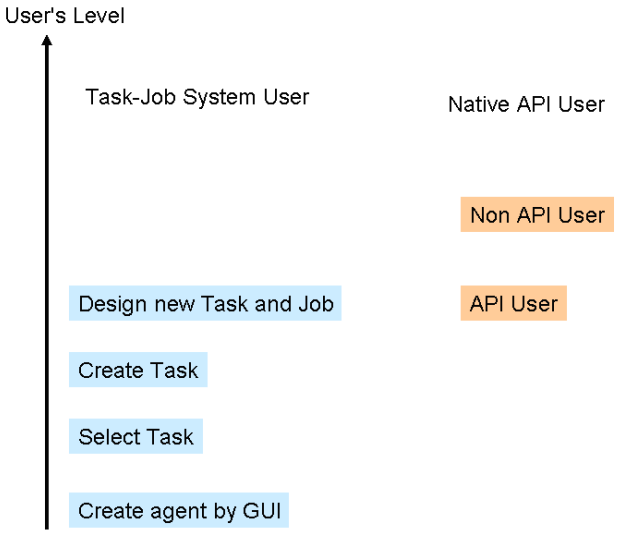

Fig. 2. Needed user level

By using this system, users can concentrate solely on improving their rescue strategies.

2.2 Job

Jobs are the smallest elements of agent action. They correspond to RCR commands such as move, rest, and clear. In addition, jobs can represent more complicated actions than simple RCR commands.

For example, 'move to fire-extinguishing position' is one of the implemented jobs. This job manages not simply the move command but also the preparations listed below.

1. Find all places (buildings, roads, nodes) that allow for extinguishing the target fire
2. Remove the burning buildings from the resulting places
3. Search for the path of smallest cost
4. Call the move command for that path

2.3 Task

Tasks are consistent action sequences. A task has a job list and achieves its goal by carrying out jobs sequentially. For instance, the structure of the task "extinguish fire A" is shown as follows:

Extinguish Task
Move Job (target=near A): move to building near A that is not burning
Extinguish Job (target=A): Extinguish fire at building A

An agent that is given this task calls the "move" job first. If the agent cannot get near A in one step, the agent performs the same job continuously (in most cases, a job requires several steps). When the agent reaches the destination, it finishes the move job and performs the "extinguish" job until the fire is under control.

The Task-Job System supports two kinds of tasks, a "system task" that is given by the system and a "user task" that is created by a user.

System Task

A system task is one that manages essential actions like extinguishing a fire or performing a rescue. These tasks are currently provided at the step prior to the user's development step.

User Task

A user task is one that manages comprehensive action. A user can freely define the jobs that make up the task and the conditions achieved by completion of the task. User tasks are adopted in complex situations that are too difficult to deal with using only system tasks.

2.4 Agent Simplicity Construction System

In our past work, we implemented the steps of "Design new Task and Job," "Create Task," and "Select Task" as shown in **Fig.2**.

Here, we design an agent construction system for simplicity (RescueBit System) using the Task-Job system by focusing on "Create agent by GUI" as shown in the figure. In this system, the user can build up an agent by using a GUI method.

Developers configure an agent's rules of action by making a command combining "Target condition" and "Task." In this method, it is possible to set several rules. The earlier a rule is set, the higher its priority is. Developers can change a rule switch (PS): If the PS is on, it is effective; otherwise, it is not. For example, we suppose that Ambulance Team Agents have the rules described in **Fig.3**. In this case, agents check the condition in the order of the rule number (Rule No.). When agents find a rule satisfying the condition, they perform the task set in that rule. However, even if the agent is able to satisfy "condition D," it does not perform "task t3" because the PS of "condition D" is off. With this system, users can develop agents by carrying out simple GUI operations.

Furthermore, we are also planning to launch a service that allows users to create agents on the Web and then to download those agents. In this way, users who are not familiar with RoboCup Rescue can easily learn to use it. Consequently, we expect this to help increase the number of first-time RoboCup competitors.

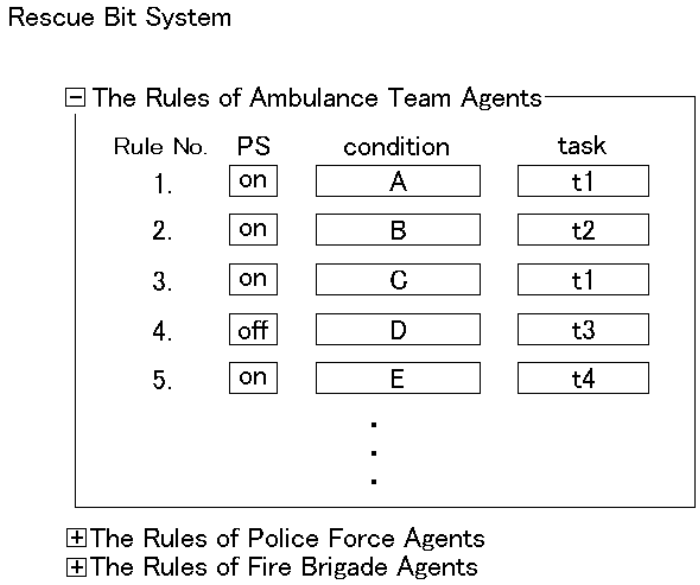

Fig. 3. Rules of Ambulance Team Agents

3 Improvement of Agents

3.1 Concept of FireSite

When agents have to extinguish fires in a number of buildings, it is important to decide the building that must be given attention first. For example, if we extinguish the fire in a building that is surrounded by other burning buildings, it will soon re-ignite.

In addition, if we try to extinguish a fire in a big building preferentially, other fires will more quickly spread because it takes a longer time to extinguish the big building's fire.

To set priorities in responding to fires, we propose the concept of FireSite. A FireSite is a cluster of buildings among which there exists a high chance of fire spreading. **Fig.4** shows an example of a FireSite, where buildings having the same color are considered to belong to the same FireSite. Here, it is easy to see that such buildings are generally adjacent to each other. When a building belonging to a FireSite is burning, the fire easily spreads to the other buildings in that FireSite. In other words, it is less likely that the fire will spread to buildings belonging to other FireSites.

When the size of a fire becomes too large, it takes much time and many agents to bring it under control. In such a case, we give up attempts to extinguish the fire and try instead to prevent the spread of fire to the other FireSites until

Fig. 4. FireSite

the immediate threat of spreading fire blows over. Consequently, this approach allows us to reduce the number of buildings that need to be extinguished.

3.2 Implementation of FireSite

As mentioned above, a FireSite consists of adjacent buildings, and thus we use outline information to detect adjacent buildings. An example of detecting adjacent buildings is shown in **Fig.5**, where rectangles "A," "B" and "C" are buildings.

Buildings next to "A" are detected as follows. First, we direct our attention to one of the building outlines as "building A," and we assume that the center of the outline is point P. Second, if the distance between P and one of the outlines of "building B" is smaller than a certain value, we determine that "building B" is next to "building A" and that it belongs to the FireSite that "building A" belongs to. By carrying out these operations, we obtain a grouping result as shown in **Fig.4**.

4 The Team's Performance

To verify the efficacy of FireSite, we ran the team agents with and without the use of FireSite and then checked the conditions of buildings.

The simulation results of our team agents using FireSite are shown in **Fig.6**, while the simulation results without FireSite are shown in **Fig.7**. **Fig.6** shows that two fires (left and upper right) were completely extinguished with FireSite.

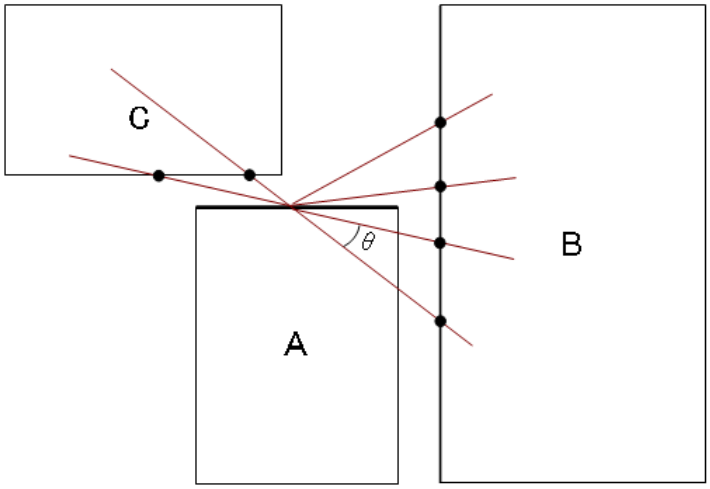

Fig. 5. Detection of adjacent buildings

In contrast, the agent team without FireSite failed to extinguish these fires. The number of buildings in each condition at the end of simulation is shown in **table 1**, which indicates good performance for the agent team with FireSite.

	Using FireSite	Without FireSite
burning	85	135
extinguished	34	6
burned out	448	638

Table 1. Conditions of buildings

By using FireSite, agents can easily predict how fires will spread. Accordingly, they can reduce attempts at extinguishing fires that have a low risk of spreading and thus focus on extinguishing more important fires.

In addition, if a fire in "FireSite A" becomes too big to extinguish, our agent team gives up the attempt to extinguish it. Then as the fire spreads from "FireSite A" to "FireSite B," the agents can extinguish the initial flames at "FireSite B" at an early stage. Consequently, we succeeded in providing a means to prevent fire from spreading from "FireSite A." Results demonstrating the efficacy of FireSite are shown in **Fig.6**, **Fig.7** and **table 1**. From this evidence, we can conclude that the concept of FireSite is an effective approach to controlling fires.

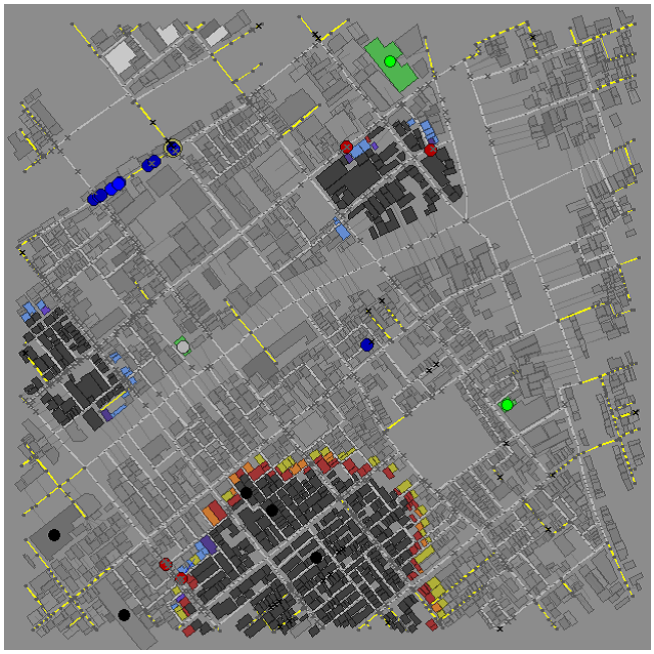

Fig. 6. Using FireSite

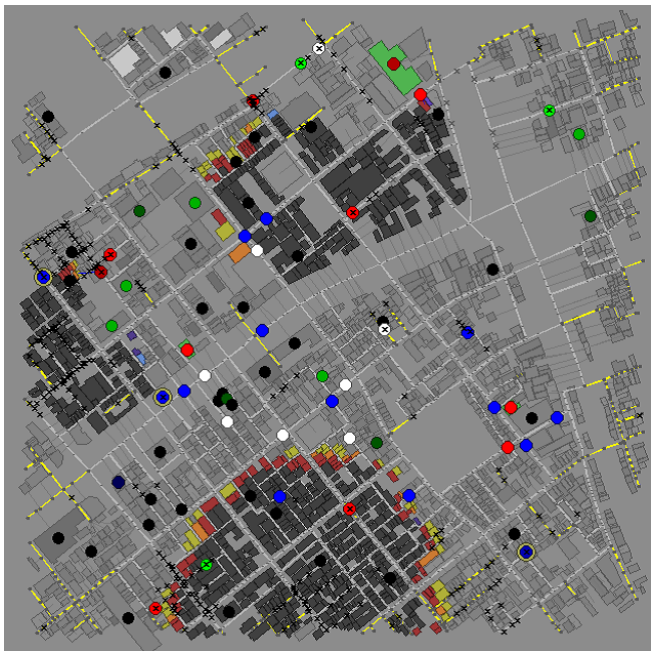

Fig. 7. Without FireSite