PAKRescueTeam's Description

Sébastien Paquet, Nicolas Bernier, and Brahim Chaib-draa

DAMAS laboratory, Laval University, Canada
{spaquet;bernier;chaib}@damas.ift.ulaval.ca

Abstract. In this paper, we describe the PAKRescueTeam project, a team of agents participating in the RoboCupRescue simulation competition. In the following, we explain all the strategies of all our agents that were used at the world competition in 2003 at Padova in Italy.

1 Introduction

It is not an easy task to develop a multiagent system to act in the RoboCupRescue environment, since it is a complex environment with a lot of challenges. To share our work on this project, we explain in this article how the PAKRescueTeam [1] team works by describing all the strategies that we have implemented for the 2003 world competition.

Even though we consider that most readers of this article already know about the RoboCupRescue Simulation environment [2, 3], we describe it very briefly to enable all people to understand it. The goal of the RoboCupRescue simulation project is to build a simulator of rescue teams acting in large urban disasters. More precisely, this project takes the form of an annual competition in which participants are designing rescue agents trying to minimize damages, caused by a big earthquake, such as civilians buried, buildings on fire and blocked roads. Our agents represent limited resources (ambulance teams, police forces and fire brigades) used to limit life and material losses. Basically, ambulance teams can rescue civilians, police forces can clear roads and fire brigades can extinguish fires. There are also center agents (fire station, police office and ambulance center) that act as communication centers.

In the remaining of this article, we describe all the strategies that helped us finish sixth out of seventeen teams at the 2003 RoboCupRescue world competition. We begin by describing the *FireStation* and the *FireBrigades*, then the *AmbulanceTeams* and *AmbulanceCenter* and finally, the *PoliceForces* and the *PoliceOffice*.

2 FireBrigade and FireStation Agents

In this section, we present how our *FireBrigade* and *FireStation* agents are working. First of all, for the *FireStation*, it is really simple, since it does not take any decision. It is simply used to retransmit messages between the *FireBrigades*

and the other centers. All decisions concerning fire extinguishment is taken in a distributed manner, locally by each *FireBrigade* agent.

The most important task of *FireBrigades* is to extinguish fires, so as long as there are fires, they will try to extinguish them. A FireBrigade takes into account all the buildings in fire. Of course, it is only the buildings that it knows about. This list is then sorted according to a certain function estimating how good is the choice of this building. The agent will then choose the building with the lowest value. The function it uses to evaluate each fire is the following:

$$\begin{aligned} f(B) = 50 * \frac{B.Fieryness}{3} - c * \frac{B.TACB}{MaxTACB} + \\ d * \frac{B.Dist}{MaxDist} + e * \frac{B.TA}{MaxTA} - 10 * nOA + \\ 5 * B.Conc + 100 * NBID \end{aligned}$$

where

$$\begin{aligned} c &= 30 * \frac{MaxTACB}{MaxTACBKobe} \\ d &= 70 * \frac{MaxDist}{MaxDistKobe} \\ e &= 125 * \frac{MaxTA}{MaxTAKobe} \end{aligned}$$

- B: The building evaluated.
- $B.Fieryness$: The intensity of the fire.
- $B.TACB$: The total area of all the 20 closest buildings.
- $MaxTACB$: The maximum value on this map for the total area of all the 20 closest buildings.
- $B.Dist$: The distance from the agent to the building.
- $MaxDist$: The maximum distance on the map.
- $B.TA$: The building's total area.
- $MaxTA$: The maximum building's total area on the map.
- nOA: The number of other *FireBrigade* agents extinguishing this building.
- $B.Conc$: Equals to 1 if the building is in reinforced concrete, 0 otherwise.
- $NBID$: Equals 1 if there is no buildings in danger, 0 otherwise.
- $MaxTACBKobe$: The maximum value on the Kobe map for the total area of all the 20 closest buildings.
- $MaxDistKobe$: The maximum distance on the Kobe map.
- $MaxTAKobe$: The maximum building's total area on the Kobe map.

All the coefficients in the last function have been tuned manually by looking at many simulations on the Kobe map. Those are the coefficients that seemed to give the best behaviors for our *FireBrigade* agents. The coefficients c, d and e have to be adjusted to be useful on different maps, it is why we adjust them according to the dimensions of the current map, using the function presented before.

With this function, *FireBrigades* are prioritizing the type of building listed here according to their corresponding coefficient:

- the early fires, with an intensity of 1, because they are generally easy to extinguish and at the border of a zone in fire;
- the fires that put the biggest area in danger, to protect the biggest area;
- the closest fires, because it is faster to reach the building;
- the smallest fires, because they are easier to extinguish;
- the fires with *FireBrigades* already on it, because it is faster to extinguish a fire in group;
- the buildings that are not in reinforced concrete, because those are harder to extinguish.

Furthermore, a *FireBrigade* agent chooses its goal fire according to the function and it will try to extinguish it until one of those conditions happens:

- the building is not in fire anymore and there is a least an other building in fire;
- the building is in fire, but the agent has made too many extinguish actions, the fires are spreading and there is a least an other building in fire;
- the building is in fire, but the agent has made too many move action, trying to reach the building, the fires are spreading and there is a least an other building in fire;

We consider the fire to spread if the goal fire has an intensity greater than 1 and if there is at least one fire with intensity 1 near the agent. This makes the agent to switch to earlier fires and by doing so, it has more chance to stop the fires from spreading.

At any moment, if the agent's tank is empty, it will go refill it at the closest refuge. The agent waits at the refuge until its tank is full. Also, if there is no building in fire, the agent will go refill its tank to be ready if a new fire is found. Finally, if there is no fire and its tank is full, the agent will search for civilians by visiting every buildings.

3 AmbulanceTeam and AmbulanceCenter Agents

In this section, we present the *AmbulanceTeam* and the *AmbulanceCenter* agents. In their case, the center has a lot of responsibilities, since it is the one making all the decisions about which civilians to rescue and in which order. We first explain how the center makes its decisions and then we explain how the *AmbulanceTeams* act based on those decisions.

3.1 AmbulanceCenter

The AmbulanceCenter agent has an important role since it chooses which civilians to rescue and in which order. At each turn, it sends the ordered list of civilians to rescue at the *AmbulanceTeams*. Those agents are described later,

but for now it is worth mentioning that every *AmbulanceTeams* are rescuing the same agent to reduce the rescuing time.

The decision process has been centralized in order to reduce the amount of messages. When a building contains many buried civilians, the messages concerning the civilians are big. This increases the chances of those messages to be lost. In the centralized decision process, if the center loses a message, the only consequence is that the civilians would not be taken into account immediately. However, in the decentralized way, each *AmbulanceTeam* may have different beliefs about civilians, so they could choose different civilians to rescue. Since all the reasoning is done with the hypothesis that all *AmbulanceTeams* are rescuing the same civilian, the reasoning process becomes invalid.

Now we will explain how the center chooses the agent to rescue. First of all, the center sends to the ambulances the order to save the other agents of the securing team (i.e. *FireBrigades*, *PoliceOffices* or other *AmbulanceTeams*), if there are some that are buried. It is really important to rescue our agents rapidly to enable them to do their tasks.

When all our agents have been saved, the center agent calculate which civilians to save and in which order. To do that, it uses a greedy planing algorithm to try to maximize the number of civilians that could be saved. Each task corresponding at saving a civilian has a time length giving the necessary time to save the civilian, taking into account the travel time to go to the civilian location, the rescuing time and the time to transport it to the refuge. Each task also has a deadline representing the expected death time of the civilian.

The Algorithm 1 presents how the first civilian is chosen. The chosen civilian is the one that maximizes the number of civilians that could be chosen after this one (*cpt* in the algorithm). When the first civilian has been chosen, it is removed form the list of civilians to rescue and the algorithm is called again to find the second one. Those two best civilians to rescue are sent to each *AmbulanceTeam* at each turn.

Even though this algorithm is efficient in most cases, we have found after the competition that in some circumstances it may not maximize the number of tasks to be executed. To illustrate that, here is an example with four tasks:

- A: $time = 3$ et $deadline = 30$
- B: $time = 5$ et $deadline = 17$
- C: $time = 10$ et $deadline = 13$
- D: $time = 10$ et $deadline = 13$

If we follow the Algorithm 1, we would choose the task A, because after this one we would still be able to execute three tasks (B, C or D) and it is better than if we choose B (one tasks), C (two tasks) or D (two tasks). However, this is not the optimal choice, since after A, we can choose between (B, C and D), but after that we would not be able to execute any task, because all remaining deadlines would have been exceeded. A better choice would have been to choose C, because after that we would have been able to do the tasks B and A.

Function FOUND-BEST-CIVILIAN(*Civilians*) return *firstCivilian*

Inputs: *Civilians*: all known civilians.
Return: *firstCivilian*: the first civilian to rescue.

```
firstCivilian ← null
maxCpt ← 0
for all x in Civilians do
  cpt ← 0
  for all y in (Civilians − {x}) do
    if x.time + y.time + CURRENT-TIME ≤ y.deadline then
      cpt ← cpt + 1
    end if
  end for
  if cpt > maxCpt then
    maxcpt ← cpt
    firstCivilian ← x
  end if
end for
Return firstCivilian
```

Algorithm 1: Algorithm used to calculate the first civilian to rescue. *time* is the rescuing time for this civilian and *deadline* is its expected death time.

3.2 AmbulanceTeam

As we have said before, *AmbulanceTeams* are rescuing civilians according to the order given by the *AmbulanceCenter*. In each message from the center, the ambulances receives two civilians to rescue. Since all AmbulanceTeam is receiving the same messages, they are always rescuing the same agent. This reduces the time necessary to rescue an agent.

When a civilian has been rescued, i.e. that it is not buried anymore, only one ambulance will load the civilian and transport it to the hospital. The chosen ambulance is the *AmbulanceTeam* with the smaller identification number, identified as the team leader. It is the same ambulance that will send a message to the center to inform it that the civilian has been rescued. All other *AmbulanceTeams* begin to work on the next civilian immediately.

When there are no civilians to rescue, AmbulanceTeams search to find buried civilians. They begin by creating a list of buildings based on the scream for help heard. Each building in a perimeter of 30 meters from the point where the message was heard are added to the list. Agents are then visiting all buildings in the list. If they found some buried civilians, they send a message to the *AmbulanceCenter* indicating the position and the healthiness of each civilian. After that, it is possible that some civilians were not heard by the *AmbulanceTeams*, so they visit all unexplored buildings to try to find buried civilians.

4 PoliceForce and PoliceOffice Agents

Polices are playing a key role in the rescue operation by clearing the roads, thus enabling all agents to circulate. Without them, some actions would be impossible because agents would be indefinitely blocked by roads' blockades. Therefore, it is really important for them to be fast an efficient.

An important aspect of our strategies is that we have divided the map in nine regions. So, at the beginning of the simulation, when the agent receives the information on the map, it begins by dividing the map in nine homogeneous regions and sends its position to the *PoliceOffice*. When the PoliceOffice has received all the positions of the *PoliceForce* agents, it assigns a sector to the nine agents that are closer to the center of a sector. Thus, there is one an only one *PoliceForce* affected to a sector and this agent has the responsibility of this sector. By doing so, we have divided our *PoliceForces* in two groups: those with a sector and those without a sector.

In the next subsections, we describe in details all the strategies, but to give an overview, here is a list of the strategies in their priority order:

1. Unblock other agents in the sector (with sector).
2. Clear roads between fires and refuges in the sector (with sector).
3. Clear roads around refuges in the sector (with sector).
4. Clear all the roads of the sector (with sector).
5. Search for civilian based on help calls in the sector (with sector).
6. Search for civilian in all unexplored buildings of the sector (with sector).
7. Clear roads between fires and refuges (without sector).
8. Search for civilian in all unexplored buildings (without sector).

4.1 PoliceForce With a Sector

First of all, the highest priority task of a PoliceForce agent with a sector is to help the other agents in its sector. This is very important, because *FireBrigades* and *AmbulanceTeams* would not be able to do their tasks if they are blocked. Therefore, when an agent is blocked it will send a message to the *PoliceForces* indicating its position. When it receives the message, the PoliceForce responsible of this sector will add it to the list of roads to clear. In this list, it will choose the closer road and will go to clear it.

When the preceding list is empty, the *PoliceForce* agents with a sector work to open the roads for the FireBrigade agents by clearing all roads between a fire in the sector and the closest refuge. The refuge can be in the sector or not. This is a proactive action to help the *FireBrigades*, because there is a good chance that those agents would ask for a road clearing. *FireBrigades* have a good chance to use those roads because they have to refill their tank at the refuge, so they are often going from a fire to a refuge and in the other direction too. This strategy helps to reduce the communications and the agent movements.

Afterwards, *PoliceForce* agents are clearing roads around refuges. They will clear all roads in a perimeter of 40 meters around the refuge, if it is in their

sector. This is also a proactive action, because refuges are intensively used by the *FireBrigade* and the *AmbulanceTeam* agents.

When the three preceding tasks have been done, *PoliceForces* clear all the roads in their sector. They first calculate the best path to visit all the roads in their sector. After this, they will follow this path and clear all block roads on their path. At the end of this task, not only all roads are cleared, but the *PoliceForce* agents have had the chance to hear buried civilians scream for help.

It is in fact the next task of the agent: search for buried civilians. They begin by creating a list of buildings based on the scream for help heard. Each building in a perimeter of 30 meters from the point where the message was heard are added to the list. Agents are then visiting all buildings in the list. If they found some buried civilians, they send a message to the *AmbulanceCenter* indicating the position and the healthiness of each civilian.

Finally, it is possible that some civilians were not heard by the *PoliceForces*. This is why the last strategy is for the *PoliceForces* to visit all unexplored building in their sector.

To summarize, after all the strategies have been done by the *PoliceForces* with a sector, all the roads are cleared and the *AmbulanceCenter* knows the position and the healthiness of all civilians, if no messages have been lost. After completing all the tasks in their sector, *PoliceForces* will act as if they had not received any sector, see section 4.2.

4.2 PoliceForces Without a Sector

The behavior of those *PoliceForces* are quite simple, they have only two tasks. The first one is to clear roads on the path from a fire to a refuge. Unlike the other type of *PoliceForces*, they are not restrained to a specific sector, so they choose the closest fire. By clearing the path from a fire to a refuge, they are clearing the roads around fires and around refuges. This is interesting since they are really important spots to clear to help the other agents to move freely in the city. When there are no more roads to clear from a fire to a refuge, they will search for buried civilians by visiting all unexplored buildings.

4.3 PoliceOffice

The PoliceOffice role is relatively simple in the simulation. First of all, they allocate the sectors for the *PoliceForces* at the beginning of the simulation. After that, they are simply redirecting messages coming from the *PoliceForces* and from the other centers.

5 Conclusion

This paper has presented the PAKRescueTeam team, a participant in the 2003 RoboCupRescue simulation world competition. We have explained all our

strategies that helped us to finish sixth out of seventeen teams at this competition.

To resume, *FireBrigade* agents are choosing the best fire to extinguish based on a function constructed empirically. *AmbulanceTeams* are always rescuing the same civilian based on the messages received by the *AmbulanceCenter*. The center chooses the civilian that maximize the number of civilians that could be saved afterwards. *PoliceForces* have a sector assigned to them at the beginning of the simulation and they are responsible to clear all roads in this sector.

We hope that this article could help some new teams in developing agents to participate in the next RoboCupRescue simulation competitions.

References

1. Paquet, S.: PAKRescueTeam web page. (Online) http://www.damas.ift.ulaval.ca /projets/RobocupRescue/index.php (Page visited on october 9, 2003).
2. Kitano, H., Tadokor, S., Noda, H., Matsubara, I., Takhasi, T., Shinjou, A., Shimada, S.: Robocup-rescue: Search and rescue for large scale disasters as a domain for multi-agent research. In: Proceedings of the IEEE Conference on Systems, Man, and Cybernetics (SMC-99). (1999)
3. Kitano, H.: Robocup rescue: A grand challenge for multi-agent systems. In: Proceedings of ICMAS 2000, Boston, MA (2000)