RoboCup Rescue 2014 – Rescue Simulation League Team Description S.O.S (Iran)

S.Mohammad Reza Modaresi, Yoosef Golshahi,
Fatemeh Pahlevan aghababa, Pegah Taheri, NavidHaeri, Salim Malakouti,
Angeh Aslanian, Aramik Markari, Morteza Rezayi[1]

[1] Robotics Research Center, Department of
Computer Engineering and Information Technology,
Amirkabir University of Technology,
No. 424, Hafez Ave.,
Tehran, Iran
modaresi@aut.ac.ir, {yoosef.golshahi, reyhaneh.pahlevan,
navid.it.haeri}@gmail.com, salimm@cs.pitt.edu,
{ angeh.a2, aramikm, m.rezayi69}@gmail.com
http://www.sosrcrs.com/

Abstract. S.O.S team has a very significant background during 12 years of RoboCup rescue simulation agent. Our team has achieved more than 8 trophies during these years. S.O.S has designed a powerful base during these years and now we are focusing only on our strategies. Our multi-layer and state-base code design helps us separate agents' strategies. Our team has been able to efficiently solve low-level decision-makings such as clearing roads, rescuing civilian or etc. since 2010, however, improving them has never been off the charts. Our strong base allows us to use, implement and test many high level strategies and intelligent strategies. New strategies have been developed in continuation of previous works. In this paper, we describe our major changes to the Fire Brigade agents. Moreover, since we find it extremely vital, later we briefly explain our low level strategies and code structure. However, we believe reading our previous team descriptions will be of extreme importance toward understanding details of our strategies.

1 Introduction

The S.O.S. basic agent and its abilities and skills have been described in previous years TDPs (2010-2012), so this paper is to depict the new strategies added to our plan for RoboCup 2014. Following will come the description of these improvements in details.

We designed new Strategies for Fire brigade and ambulance team agents, which is described in section 2. In continuation of previous strategies we developed new strategies. Most of our new strategies are related to fire brigade agents and we tried to simulate the real action of fire brigades in the real world. In section 3 we discuss our advanced agent coordination and communication methods . Section 4 describes our powerful base in details. In last section we describe our code working environment and the changes we made to customize it.

2 Agents

Reaching to reality is one of the goals of Robocop. Fire brigades have important role in Robocop rescue simulation. So the most effort was made in fire brigade. We aslo developed new state-based ambulance team and add handling aftershock to police forces.

2.1 Fire brigades

We assume that the Fire Brigade's problems and features are known to the readers so we will discuss only the technical and important parts of the Fire Brigade's implementation.

Due to limited think time (0.5 -2 sec), we should make our decisions based on some basic features and information that can be collected easily.

We run many empty simulations (with idle agents) and collect many samples from the behavior of the environment and its changes according to the agents' activity. Analyzing these samples provides important information, which is the base of Fire Brigade's strategies.

For example, surrounding the buildings by the way roads is the information that can be gained easily, but way roads don't affect fire propagation directly. Gathering many statistics about how buildings spread fire when they are near roads helped us to recognize the best way to use the relation between roads and buildings to extinguish Fire Zones effectively. (This is only a small part of useful information gathered, the rest could be found in our code)

In this year, we have tried to represent a new model by using this information, which is more similar to a real model of fire brigades than previous models.

2.1.1 Strategy

The fire brigade will have two strategy layers.

- Global view: choosing a target fire zone, and its cooperation with other agent types.
- Local view: the strategy of extinguishing the chosen fire zone effectively and the cooperation between homogeneous agents.

5 years ago, the fire brigade agents didn't use center and messages at all, therefore, we were working on full distributed strategy. As a result, other teams suffered from no communication in the last 4 maps of the 2009 final, but we maintained the same effective performance. There was a change in the agents' vision in the next year's

contest that made us update the strategy and use communications to update the world models. This change also influenced fire brigade's decision layer that changed the strategy to centralized system to select fire zones. It should be noted that this strategy performs weakly on no communication maps.

Last year our aim was to work with minimum amount of messages passed, so we made some changes in global strategy to make agents work mostly distributed and use messages only for optimizing the agents' decisions. For this aim, we used clustering algorithms with different role that will be explained later on this passage.

In the previous year, we focused on our local strategy and selected targets by distributed strategy. This year, we are working on some definition of algorithmic game theory and Nash equilibrium to select the best target for extinguishing.

2.1.2 Clustering

Clustering the map is not new for RoboCupers. Each team uses different clustering algorithms for different purposes. This year S.O.S fire group is introducing another use of clustering.

In big cities we have several fire stations. When fire happens, the nearest station takes the responsibility of extinguishing the fire. Sometimes the fire extends, which makes other stations help the busy ones.

For simulating this feature, we use star clustering algorithm and divide the map to six parts. Each part has its own agents. In fact, this strategy decreases the usage of channels in synchronization and management. It also decreases the time wasted for moving. Because the agents in one group are active, they are near each other and have similar world models. Therefore, high channel space for communication with other groups is not necessary.

2.1.3 Task Types

Associating different tasks to different agents is another feature that helps us move toward reality. As you know in real life we have different types of fire cars as in picture below.

Figure 1 Fire brigades in real world (image from http://canstockphoto.com)

In our strategy we categorize fire brigades as three different groups: Searcher, Extinguisher and free brigades.

According to our clustering strategy, we have to send agents to their best region. The first group of agents that are sent to regions are searchers, which have an important role in our strategy. In fact, they are the leader of the group that conducts sensitive tasks like checking the extinguished fire zone to determine that if any on-fire building has remained. The second group is Extinguishers that conduct their activities according to searchers' rule and the third group are free ones, which are considered backup agents to help stations when several fires happen in one region.

2.1.4 Fire probability

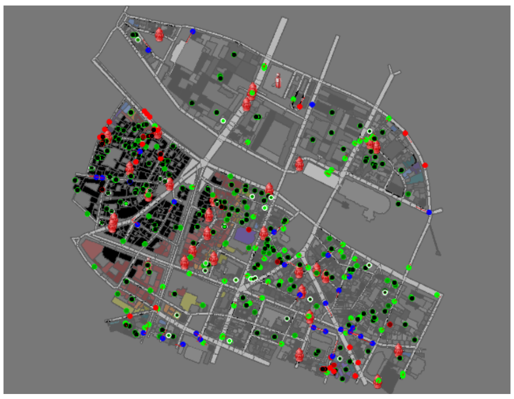
In real world, each human can estimate the fire position approximately according to the neighborhood temperature. We have tried to handle this feature in an algorithmic form. There are two ways that a building's temperature increases, one of them is the radiated energy from real neighbor and another one is the energy that air transfers to the buildings. Based on the recent start of the fire, the temperature of the air is low. Therefore, not only it is unable to increase the buildings' temperature but also decreases it because of the temperature equilibrium. Thus, we know because of the sensed temperature that one of the real neighbors is on fire.

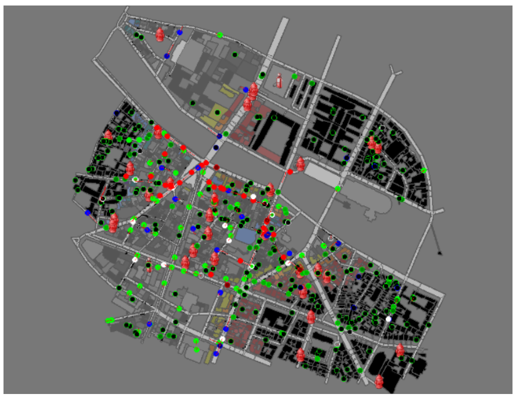
In first step, we make a graph that buildings with non-zero temperature and neighbors of them are considered as nodes and each edge has weight of zero. In second step, by using fuzzy rules like below, we try to determine the cost of the edges. The nodes that have edges with maximum cost are the most probable ones to be on-fire.

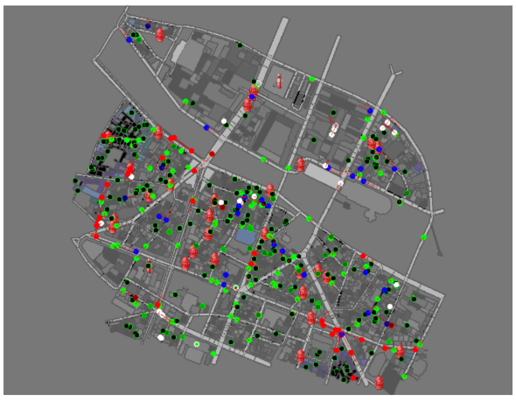
```
for each N1 in G, if hasTemp(N1) and there is N2 in NeighbourNode(N1) that
    updateTime(N2) >= updateTime(N1)  then cost[N1][N2] = -Max_value

for each N1 in G, if hasTemp(N1) and there is no N2 in NeighbourNode(N1) that
    updateTime(N2)>=updateTime(N1) then
        cost[N1][N2]=getNeighbourValue(N1,N2)*max(1,distance(N1,N2)/W1)*W2

for each N1 in G, if not(hasTemp(N1)) and updateTime(N1)<W3 and for each N2 in
        NeighbourNode(N1)

cost[N1][N2]=getNeighbourValue(N1,N2)*max(1,distance(N1,N2) /W4)*W5
```

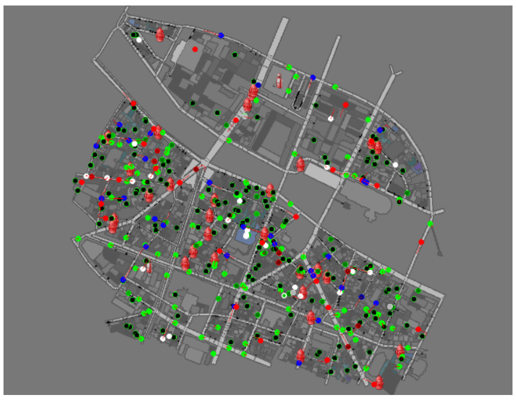
This feature was also used in robocup2013 which its results can be seen in Paris4 map on final day. The fire brigades of S.O.S estimated the real positions of fire and extinguished them.

This year, we are working on training model for this feature to better estimate fire positions by using these rules.


GUC-Paris4


MRL-Paris4


POS-Paris4


S.O.S-Paris4

2.2 Ambulance Teams

In full communication we have centralized decision and this decision is based on each AT's current task, the cost of rescuing each civilian by each AT and more other factors. According to these factors, each human is marked by an appropriate priority, and by these priorities these humans were chosen by best agent which has the minimum total cost. Because of reachability and noise, the center's decision is not trustable, so each agent first finds its own best target, and then decides between that and center's recommended target for itself. This state is called self-assigning state. In self-assigning strategy, we give a priority to each target according to agent's type or civilian, current cycle and the need of more agents in map. Target's burriedness, damage and dead time also affects the final value of priority. After that, each target's cost is evaluated by its priority and lots of other factors. Assigning algorithm works in the way to find the best AT to rescue the minimum cost target according to closer cluster and goes on finding best ATs for the targets until the current AT is chosen for a target.

Low communication strategy is similar to self-task assigning state. The difference is that we add Virtual civilian in our base code and the similar algorithm choose between humans and virtual civilians.

In No communication since ATs meet each other around the refuge the probability of going to the same cluster increases. In order to prevent this, each AT gives the higher priority to its cluster and chooses the lowest cost civilian.

Besides, in Full communication we add help and ferocious states. If the AT doesn't find a target and doesn't have search task, it will find a target with some factors in order to help others.

And the priority of helping each AT is calculated as seen bellow:

$$\boxed{\begin{aligned} Priority \; = \; & \frac{BenefitForThatAT}{MaxbenefitForThatAT} * BENEFIT\_WEIGHT \\ & + ProbablityofImportance * IMPORTANCE\_WEIGHT \\ & + \left(1 - \frac{TimeItHas}{MaxtimeItHas}\right) * LEFTTIME\_WEIGHT \\ & + \frac{\#OfCiviliansInMyCluster}{MaxCiviliansInCluster} * POPULATION\_WEIGHT \end{aligned}}$$

If there isn't any AT to help, in ferocious state the AT chooses the target considering no other condition but minimum cost.

3 Agent coordination and communication

Depending on the strategy each agent decides in a specific situation, the decision will specify whether to work centralized or distributed. However, center agents think that their platoon agents are working centralized so they provide centralized information needed by platoon agents. As the platoons have almost the same world model, their decision about this matter will be coordinated sufficiently.

As described in previous TDPs, the possible scenarios may include:

1. One or two low-bandwidth, high reliability channels and several high-bandwidth with low reliability channels.
2. A large number (10 - 20) of low-bandwidth channels.
3. One high-bandwidth, high reliability channel and a number of high-bandwidth, low reliability channels.
4. Only one channel with moderate bandwidth.
5. Only one low-bandwidth channel.
6. No radio channels at all.

Thus different strategies should be used for too specific scenarios, such as scenarios with one low bandwidth channel, low reliability channels and no radio channels. Many of these strategies were previously explained, thus, we only described the following section:

3.1 Strategy for one low bandwidth channel

For low bandwidth channel, a new kind of message blocks are used and only one message block is sent per package. This package's size is only 16 bits. At first the fire message will be sent, and then only civilians' positions are reported. The message is sent as soon as the agent considers it important regarding its priority.

3.2 Strategy for low reliability channels

We believe that, receiving important messages in the low reliability channels is more efficient than getting too many messages with normal and important priority messages and since the probability of having noise in the second packet it at most $p_f * p_f$, we decided to duplicate important messages. However, duplicating messages results in wasting bandwidth. Therefore, the messages are sent normally and the important messages are only duplicated using the aforementioned method in previous section.

4 Software Architecture

S.O.S agents are based on SOS 2009 that described in the paper we published 4 years ago. In 2010 we changed it duo to kernel changes and tried to integrate it into a newer version. We tried to find base problems and solve them. Now we have developed a collection of very good tools like reachability, message system, sensible area, fire sensible area and etc. These are the tools used for developing rescue agent strategies. The agent strategies' structure is defined as follow:

4.1 Agent design and code structure

We designed a multi-layered structure, because we believe it is easier to optimize and debug such structure. We also could divide the decision-making process of the agent to different –higher and lower- levels. Therefore as can be seen in the figure we have designed four layers.

4.1.1 High level decisions

This level chooses which state should be taken care of at the moment. It checks the state that the map currently has. It changes the priority of tasks considering the situation of the environment such as blackness and size of fire zones. This is the only part that we have been trying to train. It will let agents be flexible in different maps and scenarios.

4.1.2 States decision

For each situation we should have a plan. States are the activity of a situation that we have planned. This makes high-level decisions to decide easily and makes it easier to handle the situation without considering other situations.

4.1.3 Low level

In this level we use methods implemented in S.O.S basic agent and low level acts such as clearing a blockade. We make the decision that, in order to clear a road, which blockade should be cleared first and how should it be cleared.

5 Software Tools

We utilize eclipse as part of our IDE and Ubuntu as operating system because of its high performance and finally we use SVN for code version control. And also we provide some other tools for debugging our strategies and base.

The most important tool is Agent World Model Viewer, which provides an easy usage interface and adding layers are made so easy. Currently we have about 115 different layers that are responsible for different strategies. This is our tool for virtual debugging. The next tool is Agent Logger which logs things that happen in the code. This helps us find the problem when we are playing the logs. Other important tool that we use is log viewer. We modify the log viewer to be able to rebuild agent's world model by parsing the communication and agent sense.

6 Acknowledgements

We have great thanks to our teacher Prof. Homaioonpour and Computer Engineering and Information Technology Department of Amirkabir University for Robotics Innovation Lab, which provide us a research environment and inspires our work.

Also, we specially appreciate Mobinnet Company for their efforts, sponsorship and support.

7 References

1. A General Computational Recognition Primed Decision Model with Multi-Agent Rescue Simulation Benchmark by Alireza Nowroozi; Mohammad Ebrahim Shiri, Assistant Professor; Angeh Aslanian; Caro Lucas, Full Professor
2. Golshahi, Y et.al: S.O.S. Team Description Paper Proceeding of Robocup 2013.
3. Modaresi, M et.al: S.O.S. Team Description Paper Proceeding of Robocup 2012.
4. Markari, A et.al: S.O.S. Team Description Paper Proceeding of Robocup 2010.
5. Hashemi, B et.al: S.O.S. Team Description Paper Proceeding of Robocup 2009.

6. Ghaffuri, M et.al: S.O.S. Team Description Paper Proceeding of Robocup 2008.
7. Azizpour, H. et.al: S.O.S. Team Description Paper Proceeding ofRobocup 2007.
8. Ansari, M. et.al: S.O.S. Team Description Paper Proceeding of Robocup2006.
9. Cormen, T., Leiserson, C., Rivest.: Introduction to Algorithms MIT Press, Cam bridge(2000)
10. Horstmann, C: Object-Oriented Design and Patterns
11. R.C. DubesandA.K.Jain. Algorithms for Clustering Data. Prentice Hall, 1988.
12. Robotic Rescue Simulation league Rules.