RoboCup 2017
Rescue Simulation League Team Description
Ri-one (Japan)

Hitoshi Nakamura, Kouki Hayashi, Terumi Oguri, Haruna Iuchi
Risa Morimoto, Daimon Aoi, Ryusei Harada, Takumi Oibayashi, Takafumi Nishida
Ririko Watanabe, Kosuke Okajima, Masataka Suzuki, Eisei Mashiro

Ritsumeikan University, Japan
[is0319xx@ed.ritsumei.ac.jp]

Abstract

We used the **Gale-Shapley** algorithm for Ambulance Team, changed decision method about Fire Brigade, changed a clear method of Police Force and implemented the **A*** algorithm using *clear repair cost*of blockades. To evaluate the methods, we performed an experiment. The experimental results show that the **Gale-Shapley** algorithm outperformed the previous methods. We will explain each of these approach methods and the ways in which they are applied to the agents.

1 Introduction

The RoboCup Rescue Simulation(RCRS) is a multi-agent simulation of disaster relief activities. The RCRS server simulates various environments imitating a city after a disaster. An aim of RCRS is to make use of the virtual agents in order to rescue buried victims from under blockades, and to extinguish fires that make buildings go up in flames. Last year, our team implemented a search method using **ADACHI** algorithm, optimized a matching method which used *communication*, sharing information on buildings left out and implemented Police Force(PF) to satisfy with other agents request. Each of the chapters will describe the following contents which we implemented this year. Chapter 2 describes new clear method for PF. Chapter 3.1 describes accuracy improvement of **A*** algorithm. Chapter 3.2 describes optimization of matching algorithm for Ambulance Team(AT). Chapter 3.3 describes decision method of the position to extinguish for Fire Brigade(FB). Chapter 3.4 describes effective selection of destination about PF. Chapter 4 describes our team's score by comparing from Sample Team.

2 ExtAction

We used Point of Visibility navigation graph (**POV**) in clearing in the previous years. In the agent development framework (ADF), however, we considered that the clear method using information prepared as default alone was more efficient method than the clear method depending on **POV**. Therefore, we devised a new clear method. When a road has two neighbors and one of them is building, we defined the road as an *entrance*. We

defined the area where a PF agent was currently located as *current*, the area where the PF agent would go next as *next*, the boundary between *current* and *next* as *edge* and the vector connecting the midpoint of *edge* with the position of the PF agent as *clearline* (Figure 1). We divided the situation into two patterns when the PF clear blockades. If the *next* is not an *entrance*, the PF agent will clear along *clearline*. If *next* is an *entrance*, calculate the angle θ between *edge* and *clearline*. When θ is 90 degrees, i.e., the *edge* and the *clearline* intersect vertically, the PF agent will clear along *clearline*. If θ is not 90 degrees, by rotating *clearline* by θ, the PF agent makes *edge* and *clearline* parallel, then clear along *clearline* (Figure 2). By implementing the above method, we succeeded in clearing blockades efficiently with information given in advance from the server.

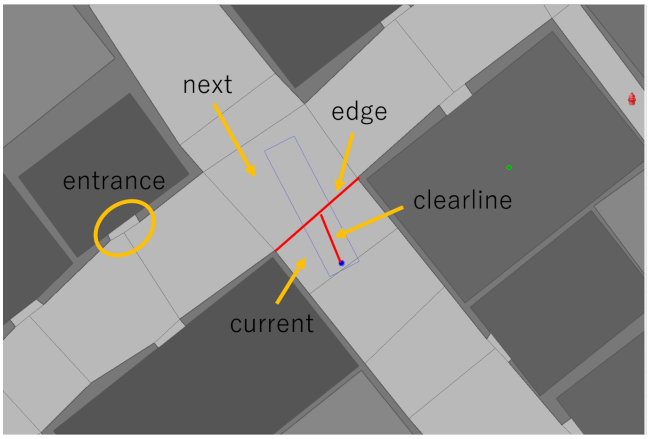

Figure 1: Definition of terms

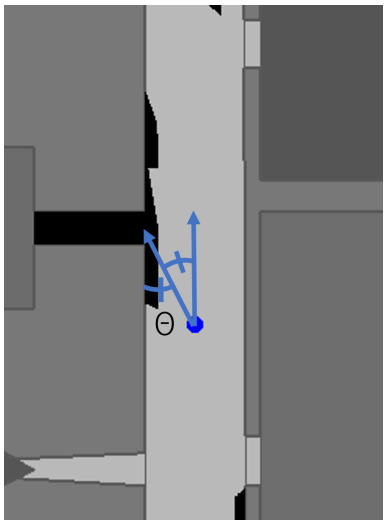

Figure 2: *next* is *entrance*

3 Modules

3.1 Path Planning

In this section, we described the modification of cost used in **A*** algorithm to improve the performance of our path planning method. **A*** algorithm is a type of route search algorithm and uses cumulated actual cost and estimated future cost for searching. We improved our method by changing the definition of the actual cost. Basically, actual cost is defined as cost of movement. In our previous work, we calculated the actual cost by using the distance of the agent moves. In the current path planning method, we added the cost of clearing blockade to the cost used in the previous work. Because of path planning with high removal cost the PF agents cleared more blockades. In addition, all agents except for the PF agents can select roads with less blockades by using the Inverse value of the cost of clearing as the actual cost.

3.2 Target Allocators

3.2.1 Ambulance Team

The main role of AT is to help victims buried under blockades, and carry them to a refuge to keep them alive as much as possible. We had employed the maximum weight perfect matching algorithm so-called **Hungarian** algorithm, for AT so far. However, this algorithm has two problems.

The first problem is that the **Hungarian** algorithm cannot applied to M to N matching problems because it is originally developed for solving N to N matching problems. In environment of rescue simulation, the number of victims is mostly larger than the number of the AT agents. That is, it is necessary to perform N to M ($N \leq M$) matching instead of N to N matching. If there are M victims, we had performed N to N matching by excluding$(M - N)$ victims.

The second problem is computational complexity. When trying to obtain a complete solution by the **Hungarian** algorithm $O(n^3)$ computational cost. Practically, the algorithm cannot finish the computation within the determined time duration. Processing was terminated by reducing the calculation amount to $O(n^2)$ so far. However, since the solution obtained by this method is not a complete solution, but an approximate solution, the accuracy of matching is not satisfactory.

In next paragraph, we propose matching by **Gale-Shapley** algorithm as a method to solve the above two problems.

Gale-Shapley algorithm is an algorithm proposed as a solution to the stable marriage problem [1]. We applied the **Gale-Shapley** algorithm to the rescue simulation(RCRS) by replacing men with AT, female with victim, and the order of desire with priority in stable marriage problem. The concrete procedure is shown below.

Input: N number of AT, M number of Victim (However, it should be $N \leq M$.)
the priority queue from each AT to Victim
the priority from each victim to AT

Output: N pair of AT and Victim

Initial state: Victim is not assigned to any AT.

Repeat steps 1 and 2 as long as there are unassigned AT.

1. AT n dequeuer the highest priority Victim m from the priority queue.
2. When AT is not assigned to Victim m, m was assigned to n.

When n' is assigned to m, which of n and n' has higher priority is judged.When priority of n' is higher than n, there are no change of assignment. When priority of n is higher than n', assignment of n' was canceled and m was assigned to n.The above is a procedure $N \leq M$, but validity will not be lost even for $N > M$.

The priority (triage) in the above algorithm is defined by the following formula.

$$\begin{aligned} triage = (Victim's\ hp\ /\ Victim's\ damage)\ -\ ((&Distance\ between\ AT\ and\ Victim \\ &+Distance\ between\ Victim\ and\ refuge) \\ &/\ One-cycle\ migration\ length\ of\ AT) \end{aligned} \tag{1}$$

Triage intended how many cycles victim can live. In brief, the lower a victim's triage, the higher his priority.

Gale-Shapley algorithm proved two things. First, if there are N to N matching, this algorithm can return complete solution. Second, if there are same the number of matching targets, computational complexity is $O(n^2)$. In consequence, it could solve the problems of the algorithm we have used. It improve precision of matching.

Nevertheless, when we use this algorithm, it causes new possibility of what AT can't be assignment completely. When start or end of simulation, it is the case where the number the number of victim which AT recognized less than the number of AT. This situation has too many the number of AT. Therefore, we used another matching algorithm for AT which didn't be assigned by **Gale-Shapley** Algorithm.

We used **Greedy** algorithm. It can select the most evaluated values.We have two reasons why we employed this algorithm. First, this algorithm can use the same priority that we employed **Gale-Shapley** algorithm. It can reduce extra calculation, because it is not necessary to recalculate the priority for **Greedy** algorithm. Second, computational complexity is $O(n)$. AT which should employ **Greedy** algorithm was done calculation which computational complexity is $O(n^2)$ by **Gale-Shapley** Algorithm. Therefore, we should employ the algorithm that computational complexity is as less as possible. Finally, we could assign victim to entire AT at all time by **Greedy** Algorithm.

3.2.2 Fire Brigade

FB aim at extinguishing fires that occurred in a disaster. It is important not to spread the damage caused by fire. Burning buildings raise the temperature of surrounding buildings. To check spread of fire, FB must extinguish the fire not from inside of a group of buildings on fire, but from outside of the group. Even in the conventional our team, we decided the place when FB extinguish fires, *extinguish position*, so that FB could extinguished from outside of the group. However, it had a problem that the determined extinguishing position was often non-optimal solution in case of insufficient to share the

information. This was because the fire condition changed every cycle, and the optimal solution also changed. In addition, as the group of buildings on fire expanded, it became difficult to grasp the whole group. Therefore, we improved the method for determining the *extinguish position*.

We decided the *extinguish position* by the following method. FB set burning buildings which existed inside the extinguishable range as Group A, and other buildings which had high temperature as Group B. In each of two groups, we formed a figure by connecting the center coordinates of the buildings. FB selected the road in the area which belonged to not Group A but Group B. FB determined the road as the fire extinguishing position. This was because that the road belonged to outside of burning building group and it was an effective position to prevent fire spread.

However, with this way, FB couldn't check the spread of fire. If the location of FB agent wasn't in an appropriate position to enclose all buildings, the fire will spread. As can be seen from the Figure 3, FB agents gathered in same place (in the yellow line) . They could not extinguish the fire in the dotted line part of blue on the left and right. Then, the fire spread further. This year, we checked the spread of fire as much as possible. FB decided the extinguishing position to surround the fire from 4 directions as shown in the Figure 4.

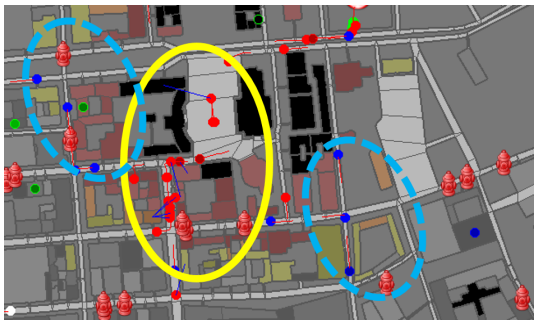

Figure 3: Bad Position

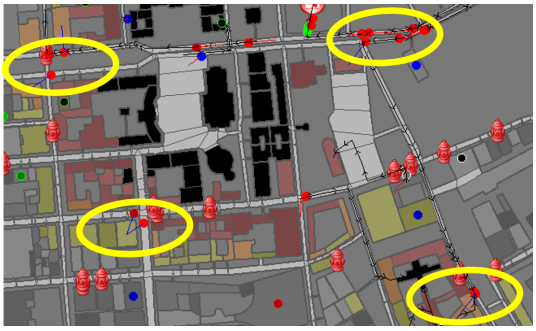

Figure 4: Best Position

The method of deciding *extinguish position* are following.

1. FB agent received the information of fire from Center Agent. FB agent could use the information of theirs visible information. The information received by *communication* to grasp the fire.

2. FB agent got the information that cluster includes the burning building. FB agent decided which cluster to manage.

3. FB agent searched burning buildings where he managed. Then, FB agent made a polygon by connecting each burning building's center of gravity as vertex. Finally, FB agent approximated it to a rectangle. Each side of the rectangle was defined

as side A, B, C and D. It is able to say that building A' is the nearest building from side A.Building A' is out of the area burning building exist. It was supposed that the road near the building A' was the best position to extinguish.

4. As same, calculate building B' , C' and D' . Then, FB agent was assigned equally from road A' to road D' .
5. If the road was crowded, the FB agent moved to road was not crowded.
6. The target building was chosen using evaluation function.

The way to evaluate building's importance is as shown below. If there was a building which a FB agent wanted to know the importance (hereinafter, this is called *BuildingX*) , The FB agent checked two kinds of influence. An inactive influence and an active influence. FB calculate the influence from considering two things. First, FB checked the temperature's rising of buildings near the *BuildingX*. Second, FB counted the number of fire buildings near the *BuildingX*. FB used these ways to selected the building which they extinguished. To use this function, FB used the *communication* to avoid duplication of targets.

In addition, we added a function of using hydrant. Before, we used only refuge. However, it had some problems. Sometimes, FB agents could not reach a refuge. There were two reasons. Firstly, blockades prevented them going to a refuge. Secondly reason, when the distance of between a refuge and the FB agent is long, it wasted time. Therefore, we used the hydrant in this year. FB agents went to hydrants when they didn't have enough water. At that time, if other FB agents used the hydrant where the FB agents wanted to supply water, the FB agents went to a refuge. This is because it was impossible to supplied water with someone in the same hydrant.

3.2.3 Police Force

The main role of PF is to clear blockades caused by a disaster. In this section, we described the selection of the targets the PF agents cleared blockades along. In the previous method, the PF agents selected, by using *communication*, the roads where other agents wanted to advance and the buildings where victims were. The method were useful partially, however; some PF agents did not clear blockades efficiently as a whole. In the current method, we proposed that the PF agents selected targets to connect important areas and cleared blockades along the targets. We regarded Refuges and *intersection*s as important areas. We defined the *intersection* as a road which satisfied two conditions: the number of neighbor is four and every neighbor is road, and next to the neighbor is road. AT goes to Refuges to transport victim, and FB uses Refuges to supply water. Most of the agents pass through the *intersections*. Since some PF agents cleared blockades along the road connecting important areas, those PF agents opened the main road early in the simulation.

4 Finary Results

We compared the score of this year's program from SampleTeam results and Preliminary results. From the following table,we can improve our program a little.

Team	**Map**				
	Kobe1	**Berlin1**	**Istanbul1**	**VC1**	**Eindhoven1**
Ri-one	4.78	24.09	46.15	5.88	73.36
Preliminary Results	4.50	24.57	61.12	5.88	46.86
SampleTeam	4.15	24.42	44.20	5.82	81.74

References

[1] KokosugakunoUtsukushikimonogatari“AnteimattingutoGale-Shapleyarugorizumu” http://mathtrain.jp/galeshapley (accessed 2017-12-23)

[2] Akira Kinose, Futoshi Sato, Tatsuki Fukumori, Yukino Watanabe, Ryusei Nakatsu Masataka Yamaguchi, Yuki Kobayashi, Hitoshi Nakamura, Kouki Hayashi RoboCupRescue 2016-Rescue Simulation League.Team Description.Ri-one(Japan), 2016.