RoboCup 2018 – TDP Rescue Agent Simulation Ri-one (Japan)

Takafumi Nishida, Kosuke Okajima, Haruna Iuchi, Eisei Mashiro, Masataka Suzuki, Takumi Oibayashi, Risa Morimoto, Soichiro Komura, Akinori Kanechika, Hiroaki Nozaki, Keita Fukui,Takahiro Ijichi, Terumi Oguri, Kaito Masuda, Yamato Higashi, Joichiro Amada, Haruki Nakamura

Ritsumeikan University, Japan
[sc0072pe@ed.ritsumei.ac.jp]
http://rione.org/hp/

Abstract. We implemented new methods in modules: reduction in a number of nodes required for calculation was implemented when agents calculated paths; we implemented a new path planning which made agents go to roads where agents could pass; *K-means++* algorithm was implemented for clustering instead of *K-means* algorithm.

We implemented new strategies: efficient assignment of ambulance teams (AT) for victims according to a dynamic changing condition of AT was implemented; fire brigades (FB) were allocated to dynamic clusters of fires; an algorithm which could determine whether to use refuge or hydrant to supply water efficiently was implemented for FB; a method of searching for fire which could adapt to all situations was implemented for FB; a method of clearing blockades along an edge of a road for police forces (PF) was implemented; an algorithm which could choose roads prioritized by several things was implemented for PF. To evaluate the method an experiment was performed.

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 virtual agents to rescue buried victims from under blockades and extinguish fires that make buildings go up in flames.

Last year, we implemented some strategies: the Gale-Shapley algorithm was used for AT; a method to decide FB's targets was modified; PF's method to clear blockades was modified and the A* algorithm using cost to clear blockades was implemented[1].

During this year, we implemented new methods in modules and strategies. Each of the chapters described the following contents which were implemented this year. Chapter 2.1 described an accuracy improvement of *K-means* algorithm. Chapter 2.2 described a new route search method. Chapter 3.1 described an improvement of matching algorithm for AT. Chapter 3.2 described three new

methods for FB. Chapter 3.3 described a method of determining a target for PF according to priority. Chapter 4 described a result of an experiment. Chapter 5 described conclusion of this development.

2 Modules

2.1 Clustering

Last year, *K-means* algorithm was used to divide a whole map into clusters so agents could do rescue activities efficiently in our team. However, improper points were chosen as initial centers of each cluster in a map which had the biased distribution. Therefore, *K-means++* algorithm was implemented for clustering instead of *K-means* algorithm[2].

In *K-means* algorithm, initial centers were chosen from all points in the map with equal probability. Therefore, if improper points were chosen as initial centers, the solution could be worse. *K-means++* algorithm solved this problem. In *K-means++* algorithm, initial centers were chosen spreadly as possible. Specifically, the probabilities were defined by below steps.

1. Choose one center uniformly at random from all points.
2. The probability is defined by the following.
3. New initial center is chosen by this probability.

$$P(X_i) = D(X_i)^2 / \sum_k D(X_k)^2$$

$P(X_i)$: the probability X_i is chosen for initial center

$D(X_i)$: the distance from the nearest center to X_i

4. Repeat those steps until k centers have been chosen.

The initial centers to be away from each other were chosen frequently, and initial centers were chosen according to the probabilities. As a result, the improper points were chosen only occasionally in a biased map. (Figure 1,2)

Fig. 1: Results by K-means

Fig. 2: Results by K-means++

2.2 Path Planning

2.2.1 Route search using intersections

Using a path planning before improvement, agents searched for the shortest route from agent's current location to a target. Adjacent areas were scanned from the agent's current location until it reached a target. However, this calculation took a long time to complete. Therefore, we implemented a reduction in a number of nodes required for calculation by using an *intersection* as a *node* and a road between the *intersections* as an *edge*.

If there were no branching routes from agent's current location to a target, a result of a calculating path didn't change although the calculating path was repeated. In other words, the result of calculating path depended on the number of *intersections* in the route from agent's current location to a target. First, a

simplified graph was created with *node*. We use a depth-first search algorithm to create a graph. There is only one way between two mutually adjacent intersections, so if we continue the search while judging whether it is an intersection or not, we will definitely reach the next intersection.

Then, an *intersection* was defined that three or more adjacent areas were wholly roads, and neighbors are not adjacent to a building, and *edges* were managed as a list. Figure 1 shows a graph created by roads. Figure 2 shows a graph created with *intersections*. We use the A* algorithm for path planning. When calculating path planning, it is optimal because there are a plurality of destination candidates.

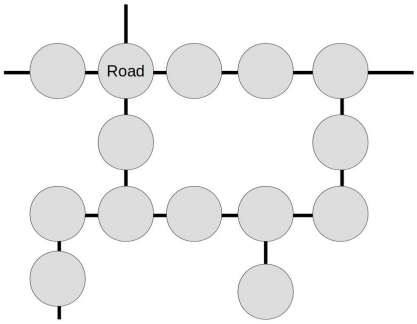

Fig. 3: A graph created with roads

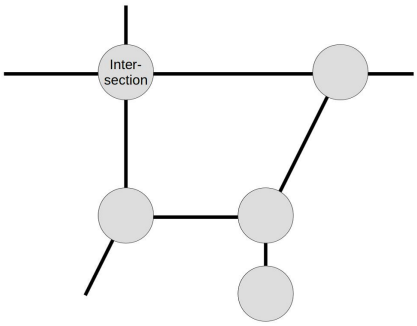

Fig. 4: A graph created with intersections

2.2.2 Path planning to avoid the blockades

In our conventional path planning, there was a problem that a path was not generated in consideration of a state of blockades. Therefore, we implemented a new path planning to avoid the blockades. First, we defined *AREA* as "java.awt.geom.Area". By this definition, we could define the shape of blockades. When a path was generated by the new path planning, agents used the shape of recognized blockades.

If there were recognized blockades in the coordinate through which agents were going to pass, they judged that the road was impossible to pass. The new path planning made agents go to roads where agents could pass. When a shape of the blockade was gotten, the computational complexity was reduced by expanding the recognized areas concentrically.

3 Strategies

3.1 Ambulance Team

The main roles of AT are to rescue victims buried under blockades and to carry victims to refuges. It is needed for efficient rescue in the rescue activities to make a *cooperation* sharing the purpose of an action. Therefore, the *cooperation* was investigated in order to find how *cooperation* could rescue more victims. We implemented an efficient assignment of AT for victims according to a dynamic changing condition of AT.

The logs of finial of RoboCup2017 Rescue Simulation League were analyzed by paying attention to rescue victims[3]. *Rate of rescued victim*, *a rate of rescue action* and *a rate of cooperation* were calculated from this analysis. *Rate of rescued victim* was a rate by dividing the number of rescued civilians by the number of civilians who needed a rescue. *Rate of rescue action* was a rate by dividing the number of cycles which AT rescued victims in by the number of all cycles. *Rate of cooperation* was a rate by dividing the number of rescue with *cooperation* by the number of all rescue. The rate of cooperation is calculated according to the ratio of the number of rescue teams rescuing the same target among the number of rescues of all steps. Table 1 shows the data of the logs. Table 2 shows the results of analyzing the data of the logs.

Table 1: Data of the logs

Map	VC3
Number of agents	50
Maximum number of action step	319
Number of action	15950
Number of victims	306

Table 2: The results of the top team log data analysis in 2017

Team	Rate of rescued victim	Rate of rescue action	Rate of *cooperation*
Aura	10%	20%	23%
MRL	22%	17%	22%
RoboAKUT	13%	18%	21%
SEU-UniRobot	9%	22%	19%

From the result of the analysis, MRL could rescue many victims, because MRL built groups of AT dynamically and reduced time to rescue victims. In contrast, SEU could not do an efficient assignment, because the *rate of rescue action* of SEU was high, but the *rate of cooperation* of SEU was low.

Last year, we used *Gale-Shapley* algorithm for this problem. *Gale-Shapley* algorithm was suggested as a solution for the stable marriage problem. To replace men with AT, women with victims and an order of desire with priority was able to adapt to RCRS matching them on one-to-one. However, the result of analyzing cooperation of rescue taught a fact: quick rescue depending on the collective action. The situation many AT rescued a victim could save more victims. Therefore, an algorithm to do group action dynamically was made.

The concrete methods are following.

Inputs: define "N" as a number of AT, define "M" as a number of a victims, queue equipped priority for victim from AT

Output: define "L" as a number of groups of pair of AT and victims

Initial state: any AT weren't assigned to victims

If there were AT having not been assigned and elements of queue equipped priority for victims, operated 1 and 2 were repeated.

1. AT(N) picked up the victim(M) from queue equipped priority and selected the victim as a candidate.
2. Three of AT existing in a circle with the radius of 50 selected in the order of distance from a victim(M). If AT(N) was contained in the group made of three AT, M was assigned to N.

The above was in operation when a victim(M) is more than AT(N). The validity was not lost when AT(N) is more than the victim(M). Therefore, the algorithm of group action could rescue more victim than *Gale-Shapley* algorithm of 1;1 matching because of a group of building and reduce time to rescue. However, a problem that there was a possibility AT wasn't assigned victim occurred in the case of the algorithm of group action. Accordingly, the *greedy* algorithm also was used to assign victim to AT don't have a target. Table 3 shows the difference between last year's log data and current log data.

Table 3: Ri-one's log data analysis result.

Team	Rate of rescued victim	Rate of rescue action	Rate of *cooperation*
2017 Ri-one	0.99%	14.35%	1.02%
2018 Ri-one	1.48%	10.87%	3.62%

3.2 Fire Brigade

3.2.1 Method to determine targets

FB aimed to extinguish a fire occurred in a disaster. It was important not to spread damages caused by the fire. Therefore, it was necessary to extinguish the fire as soon as it occurred. However, the number of FB was limited, there might be multiple locations occurred fire, or the situation of the disaster was changing with every moment. These were problems. Because of them, FB had to be allocated according to the situation of the fire.

Thus, FB were allocated to dynamic clusters of fires.At this time, the number of the allocated FB was changed according to the cluster 's size, number of all FB. Last year, FB were allocated to static clusters of buildings. This didn't take the state of the fire which changed every moment into consideration, therefore FB's movement may be wasteful and FB may be short. Dynamic clustering was implemented to solve the problems.

We implemented method of dynamically generating clusters and updating them as follows. A fire station collected information of burning buildings from messages by agents and processed them. A fire station also checked the list of clusters. If the list had no elements or the disaster had been reported far from all of the clusters' basis, a new cluster was generated making the burning building that had been collected as a basis. If a burning building was detected within a certain distance from the center of the cluster, a fire station contained its entity into the cluster. FB selected a target with using those clusters.

Updating clusters with its central coordinates made possible that plural clusters had the same building in their own area. Besides, disaster spread from first-generated disaster source. Consequently, this method was implemented that make one building as a basis of the cluster.

3.2.2 Method to supply water

Efficient extinction and supplying water are essential to FB. Therefore, we paid attention to efficient use of hydrants. Last year, hydrants weren't really used in our team. It was because hydrants had a lower quantity of water supply per time than a refuge. In addition, if multiple agents supplied water in the same hydrant, the quantity of water for each of FB was decreased. It wasn't efficient way. However, sometimes agents could return to the site on fire earlier by using hydrants than using the refuge.

We implemented an algorithm which could determine whether to use refuge or hydrant to supply water efficiently. Specifically, at first, FB calculated distances from agent's current place to the closest refuge and distances from agent's current place to the closest hydrant. If the refuge was closer than the hydrant, FB with no water went to the refuge. If the refuge wasn't closer than the hydrant from the agent's current place, the algorithm calculated a difference of arrival time by using the distance from agent's current to the closest refuge and the distance from agent's current place to the closest hydrant. The quantity of water

the agent could supply in time of the difference of arrival time was calculated by algorithm. From agent's current place, FB determined whether to use refuge or hydrant to supply water efficiently. At this process, hydrants which other agents were going to use or was supplying water were ignored by the algorithm. It was because if multiple agents tried to go to the same hydrant, only one fire brigade had to supply water there. The other FB had to go to the others hydrant.

3.3 Police Force

The main role of PF is to clear blockades which are caused by a disaster. In particular, to ensure traffic routes for other agents such as AT and FB are PF's top priority. Hereinafter, an *entrance* is defined as a road adjacent to a building.

3.3.1 ActionExtClear

Last year, PF cleared blockades as they went to a center of the road. However, with this way, if PF didn't go to the center of the road at all times, they got caught in the projections.

Therefore, PF cleared blockades as they went not to the center of the road but along an edge. Consequently, projections in which agents were caught were not generated and PF could clear blockades at an entrance easily. Therefore, they got easier to rescue other agents.

3.3.2 Detector

In the case, information which PF obtained is reflected as a target for the detector, the target was considered to be comprehensively selected. In concrete, the hashmap was implemented, inserted a priority as a value into using each EntityID of all roads as a key.
The priority corresponds to the following 7 items
・*Entrances* of refuge
・Roads regarded as an *intersection*
・Roads on which AT or FB are in the blockages
・Roads in a cluster
・*Entrances* of the buildings in which humans are buried
・Roads close to burnt-out buildings
・Roads without blockades

As with last year, both refuges and *intersections* were considered to be the most important areas because AT and FB often went to refuges and *intersections* were areas which had a high traffic volume. The priority of an *entrance* of the refuge was increased and the priority of road regarded as an *intersection* was increased. In this chapter, an *intersection* was defined that it had 4 neighbors which were roads and 2 neighbors away from it were 2 roads.

It is necessary to secure routes for AT and FB which have been caught in blockades since a simulation starts. Thus, the priority of roads was increased

in the case AT and FB were caught in blockades on the road. The priority of AT and FB were increased, but FB who reduced the spread of damage were considered to be more important than AT.

It is inefficient that several PF clear the same road. Therefore, the priority of the road in a cluster was increased to share work sections. Additionally, the priority of the *entrance* of the buildings in which humans were buried was increased.

Few civilians were living around burnt-out buildings and burn-out buildings were surrounded by burnt-out or almost burnt-out buildings. For this reason, there was not much traffic and the priority of the road of neighbor was decreased.

EntityIDs of roads which had no blockades were stored in a set named clearedRoads in advance and the priority of a road was removed from the hashmap. In the case aftershocks occurred, the set was initialized because of increasing amount of blockades around.

In addition to above 7 items, the priority of value was determined by a type of message received. In the case the message received was about roads, the road which was passable was added to the set named clearRoads.

In the case the message received was a message about AT, if AT's action was ACTION_RESCUE or ACTION_LOAD, it was determined that the AT were in a building and a few blockades were at the *entrance*. Therefore, the priority of the *entrance* which the AT were in was decreased. In the same case, if AT's action was ACTION_MOVE and the target was a human who was in a building or a building, the priority of the *entrance* was increased.

In the case the type of message received was a message about FB, if FB's action was ACTION_MOVE and the FB's target was a road, the priority of the road was increased.

In the same case, if FB's action was ACTION_REFILL, the FB was in a hydrant. Then, if an FB's target was building, the priority of the *entrance* of it was decreased and if FB's target was a road, the priority of the road was decreased.

In the case the message received was about PF, if two PF selected the same target, either result was assigned null.

In the case the message received was about PF's command, the priority of the road indicated was increased.

4 Conclusions

Following is the conclusions of this paper.

- A buried civilian came to be rescued by two or three AT.
- FB were clustered dynamically taking account of current fires, and they became to be able to supply water at a proper point.
- AT and FB learned to move to a destination while avoiding blockades.
- PF were improved efficiency by more depending on obtaining information and clearing blockades along a road.

References

1. 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
RoboCupRescue 2017-Rescue Simulation League. Team Description. Ri-one (Japan), 2017.

2. Takashi Onoda, Miho Sakai, Seiji Yamada, "Comparison of Clustering Results for k-means by using different seeding methods", 27th Fuzzy System Symposium,2011.

3. Keisuke Kubota,Kazunori Iwata,Nobuhiro Ito,"A Consideration on Cooperative Behavior of Disaster Rescue Agent." ,SIG-SAI the 29th Study Group,2017