RoboCup 2017 – Rescue Simulation League Team Description <Apollo (P.R. China)>

< Jiedong Yang, Ziang Li, Zhiping Chen >

Nanjing University of Posts and Telecommunications, China
Apollo Robotics Innovation Lab, NUPT
yangfre1up@gmail.com
http://apollo.icecoffee.cn

Abstract. This paper describes the main features of the Apollo-Rescue rescue simulation team that is going to participate in the RoboCup 2017. In the past, a lot of work has been done in our code based on the new ADK framework. In this paper, an improvement of clustering will be introduced and a new path planning method for platoon agents and some implements in target allocators will be covered.

1. Introduction

It's the second time for Apollo to participated in RoboCup Competition. We've achieved a 4-th place in RoboCup 2015. Since then, we made an annual conclusion, found some major problems during other competitions like Iran-Open and tried to improve the performance and overcome the weaknesses of our code.

Based on the new ADK framework, we improved the clustering procedure for agents, using a new path planning method for platoon agents, improved the target allocator for different types of agent. By all means of these efforts, we expect to become a high-quality rescue team in the near future and achieve better performance in 2017 competitions. More details will be introduced below.

2. Clustering

Clustering is the fundamental element for agent simulation. During the simulation, there are two kinds of the situation need to be solved. The first one that we need to think about is how to manage to make all agents work efficiently and try to avoid the situation like same area being checked twice or even more. The second question is to depict the burning buildings which adjacent to each other as one closed space instead of seeing them as a single building that needs to be extinguished.

To solve the first question, we decided to use K-Means algorithm and divide map into many different areas. And make every agent have a partition assignment for themselves. And about the second problem, the convex hull seems to be a good solution for it. By using Jarvis's march algorithm, the burning buildings can be depicted as a convex hull quickly and efficiently.

2.1 K-Means Algorithm

We use *K*-means algorithm to divide the map. According to the amount of PF, we divide the map into several partitions. At the beginning, we randomly choose several points as initial points. Then we accord to this formula to calculate all buildings' distance.

$$c^{(i)} = \arg\min_{j} \| x^{(i)} - u_j \|^2 \tag{1}$$

Then we use the second formula to calculate the average of the distance between the original data and the original data.

$$u_j = \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}} \tag{2}$$

u means the position of the building and ***x*** means the position of the selected entity. And then iterate two formulas, until all of the U is not almost changing.

2.2 Jarvis's March Algorithm

The idea of Jarvis's Algorithm is simple, we start from the leftmost point (or point with minimum x coordinate value) and we keep wrapping points in a counterclockwise direction.

Following is the detailed algorithm.

1) Initialize p as leftmost point.
2) Do following while we don't come back to the first (or leftmost) point.
   a) The next point q is the point such that the triplet (p, q, r) is counterclockwise for any other point r.
   b) next[p] = q (Store q as next of p in the output convex hull).
   c) p = q (Set p as q for next iteration).

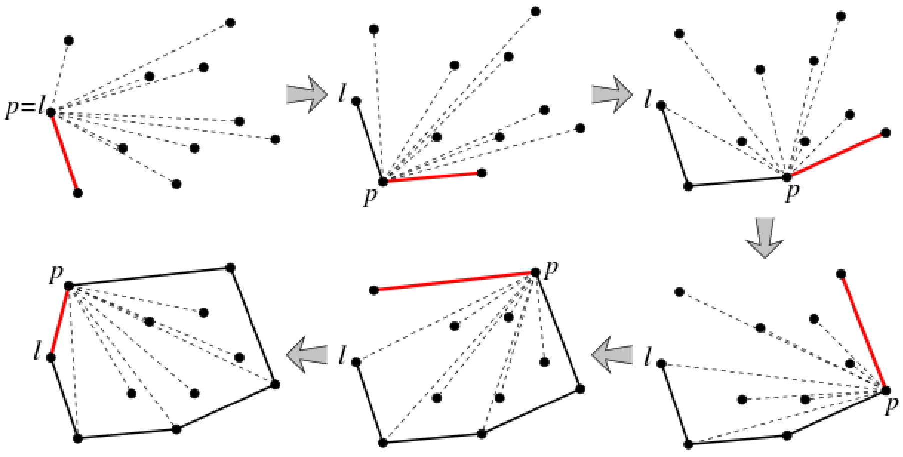

Fig.1 The execution of Jarvis's March

After the convex hull is complete then we can define a set of edges and a set of apexes. These two sets can really help fire brigades making a better target selection in the extinguish job. For example, the fire brigade agents will choose the burning building which is near the edges or apexes over the one that not. Because the outer one is dangerous than the inner one.

3. Path Planning

The purpose of path planning is to help agents find a path leading to the agents' target. A* (pronounced as "A star") is an algorithm that is widely used in pathfinding and graph traversal, the process of plotting an efficiently directed path between multiple points, called nodes.

The core formula of A* algorithm is:

$$F = H + G \tag{3}$$

G = From the starting point A, along the path generated, moved to the grid on the specified grid movement costs.

H = Moving the grid to the end of the B from the grid to estimate the movement cost.

Following is the pseudo code of A* algorithm.

```
1. Initialize the open list
2. Initialize the closed list
    put the starting node on the open list (you can leave its f at zero)
3. while the open list is not empty
   a) find the node with the least f on the open list, call it "q"
   b) pop q off the open list
   c) generate q's 8 successors and set their parents to q
   d) for each successor
```

```
        i) if successor is the goal, stop search
         successor.g = q.g + distance between
                    successor and q
         successor.h = distance from goal to
         successor (This can be done using many
         ways)

         successor.f = successor.g + successor.h
       ii) if a node with the same position as successor is in the OPEN list which
           has a lower f than successor, skip this successor
       iii) if a node with the same position as successor is in the CLOSED list which
             has a lower f than successor, skip this successor otherwise, add the node
             to the open list
    end (for loop)

    e) push q on the closed list
    end (while loop)
```

By using this algorithm, agents can find their path more efficiently. The method is helping police force on their clearing efficiency. With the ability to clear blockades and a smarter path planning method, the random blockades located on the road can be removed quicker than it was.

4. Target Allocators

The major difference between the new ADK framework and the old one is that the center agent is in charge of assigning tasks. So the target allocators become a very critical method regarding the rescue action and cooperation. In this section, we'll introduce how we implement the target allocator.

4.1 Target Allocator for Ambulance Team

First, AT agents are now controlled by a central intelligence agent. So the way we select our target is different from past times. To solve the problem of MAS (Multi-Agent System) task allocation, we do not need to select the complex mathematical tools to establish the appropriate application model. Secondly, we will describe and measure the status and function of each member agent in the process of accomplishing the common task, and then the tasks such as task allocation and resource allocation The Game theory is recognized as an effective mathematical tool for studying human social interaction. It takes the utility function as the core and pursues the maximization of utility.

The following is a pseudo-code：

```
Enter: resources, perform Agents;
Output: The result of executing Agent bid resources.
{
```

```
Initialize the resource matrix, execute the Agent matrix;

WHILE (resource not empty) {
(1) FOR (i = 1; i <= n; i ++)
      Execute Agent i to sort the resources agent want to purchase by priority
      for $r_i^1, r_i^2, \ldots, r_i^{mi}$
(2) FOR (i = 1; i <= n; i ++) {
The executing agent makes a request to resource $r_i^1$;
IF within the specified time, $r_i^1$ did not receive other requests to perform
  Agent, $r_i^1$ is assigned to the implementation of Agent i
ELSE {
  The execution agents that calculate the competition $r_i^1$ obtain the next major
  profit of the resource $W'$ $(including: W_m' W_n' W_{p\ldots}')$;
  Calculate the performance of the competition $r_i^1$Agent to obtain the re
  source's profit rate of change (such as : $\eta_m$=( $W_m$-$W_m'$)/$W_m$ , $\eta_n$=( $W_n$-
  $W_n'$)/$W_n$ ,$\eta_p$=( $W_p$-$W_p'$)/$W_p$ ;
If the maximum rate of change is different, $r_i^1$ is assigned to the agent with
    the largest rate of change.
Otherwise, allocate $r_i^1$ to the execution agent with the largest increase in
execution capability;}}
(3) resource management agent to reorganize the resource matrix;}
return the resource purchase results; }
```

In the actual situation of RoboCup rescue simulation, we define the distance from AT agent itself to the civilization or other agent need to be rescued as the “resources”. We tried to consider about the HP, the damage, but most of the time we can’t obtain these information, so if only in extreme condition, like a civilization is about to die in next few time steps. Or we will only take distance into consideration.

4.2 Target Allocator for Police Force

We use `Greedy Algorithm` to allocate tasks for PF. At first, we define different weights for different situations. For example, as we have seen, the most important situation is the fire brigade who has been blocked(shown in figure 2). But if the PF sees an agent blockade, he will save him at first.

```
BLOCKED_FIRE_BRIGADE(250),
REFUGE_ENTRANCE(179),
FIERY_BUILDING_1(190),
BLOCKED_AMBULANCE_TEAM(187),
BUILDING_WITH_HEALTHY_HUMAN(186),
FIERY_BUILDING_2(185),
FIERY_BUILDING_3(180),
BURIED_FIRE_BRIGADE(177),
BURIED_POLICE_FORCE(175),
BURIED_AMBULANCE_TEAM(140),
BUILDING_WITH_DAMAGED_CIVILIAN(110),
SEEN_AGENT(1000), DEFAULT(1);
```

Fig.2 Weight for Different Situations

At the beginning, we allocate several tasks for each PF. When they begin to act, they will refresh task lists constantly. They will only finish their own partition's tasks. We define an algorithm to calculate the final weight to judge which is the most important one to act. Our algorithm is related to the weight of situation and distance.

$$\boldsymbol{F} = \boldsymbol{W} + \boldsymbol{D} \tag{4}$$

The ***F*** means the final weight, ***W*** means the beginning weight we defined and ***D*** is the distance-weight we used in the algorithm. We use the distance between target and agent and world time to calculate ***D***.

$$\boldsymbol{D} = \boldsymbol{K} * \textbf{worldTime / distance} \tag{5}$$

If a target is too far away, we hope its weight should be low. And if the time is close to the end. We don't hope the PF will move too far away. So we use the ***F*** to decide which target should be cleared.

4.3 Target Allocator for Fire Brigade

To improve the efficiency of extinguishing there are some major problems need to be solved. The first problem needs to be solved is the target locating. A lower fire level building usually located near the edges or apexes of the fire buildings' convex hull. So we selected them as our targets and put all of them in one list. Fire Brigades' actually includes two step: 1. Target selecting 2. Task allocation. Our work is more focus on the first step. Like we said before, the target selecting is simply taking the building has lower fieryness as target selection. It works at when fire zone is single or small and seldom. But it's not the proper method for the situation like there are multiple fire zones. So to solve this situation, we decided to add a method to improve the extinguish performance on multiple fires.

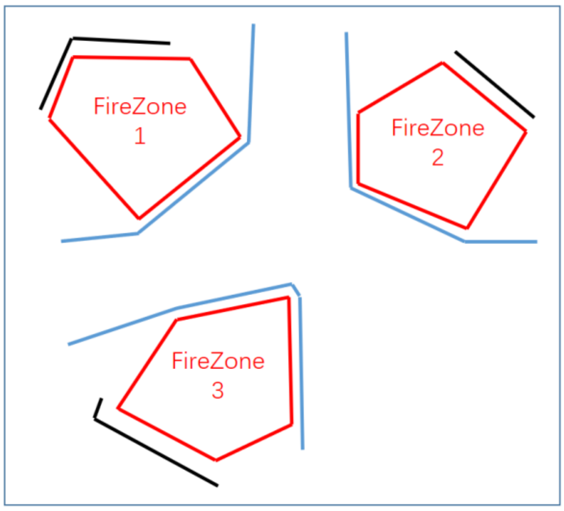

Fig.3 Illustrate of target selection about multiple fire zone

As what figure 3 showed, the blue line is the side we prefer to choose when extinguishing multi-fire situation. And the black line is the side we prefer agents not to go to. We provided a simple method to approach this. As you can see there are gap areas between the fire zones, then we divided this area into 3 partitions, and calculate the proportion of burning buildings in total buildings. Using N as the number of fire zones, P as the number of buildings in one partition, F as the number of burning building in one partition and $\eta = F/P$. So we select the buildings has fieryness from 0 to 3, then divided them into N partitions, each partition is preserved in every single list. And then counting F and P, calculate η. The principle of selection is choose the one with more burning buildings and lower η value. Then the burning building inside this partition has the higher priority. This will generate the blue line in a very simple way.

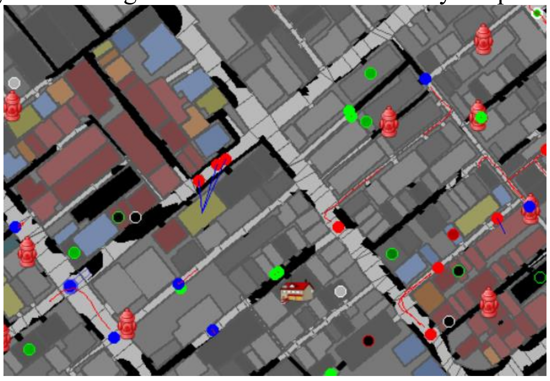

Fig.4 Fire Brigades works in between the fire zones

About this method, the thinking of extinguish strategy could be a little different when it involves the multiple fire situation. Instead of seeing them as separated areas but considering protect the gap area from fire. About the task allocation, they are quite same with the ambulance teams' strategy.

5. References

1. Carmen Gervet and Slim Abdennadher. Multi-agent planning for the robocup rescue simulation – applying clustering into task allocation and coordination. In international Conference on Agents and Artificial Intelligence (ICAART), pages 339-342, 2012.
2. Benjamin Balaguer, Stephen Balakirsky, Stefano Carpin, and Arnoud Visser. Evaluation maps produced by urban search and rescue robots: lessons learned from robocup. Autonomous Robots, 27(4):449-464, 2009.