RoboCupRescue – Rescue Simulation League

Team Description

<ZJUBase (P.R.China)>

Institute of Cyber-Systems and Control
Zhejiang University, P.R.China
leijinghao@yeah.net

Abstract: In this document, we describe some specific features of ZJUBase rescue simulation team applying for RoboCup 2014. Model building and algorithm implementation of the main aspects of rescue simulation are discussed. Flexible software architecture, multi-mode path planning, self-adaptive communication model, fire-predictor, agents strategy and partition algorithm are depicted in detail.

Keywords:RoboCup, RCR(Rescue Simulation League), road classification, multi-agent AI, convex hull

1.Introduction

The RoboCup Rescue Simulation environment is of great social value and it provides a platform for the development of algorithm design, artificial intelligence, statistical learning and data mining. To achieve a better performance, a hierarchical agent structure is established in our implementation. Communication model is designed in steps to ensure that messaging among agents is auto-controlled and relatively reliable. We implement A* algorithm and some other shortest path algorithm based on the abstract universal graph {V, E}. Clustering with connectivity repair function is used in the partition of the map. Components and other architecture structures are used so that the source codes are loosely coupled.

2.Software Architecture and Tools

Our architecture gains more flexibility, extensibility, testability and reliability, and reduces the coupling between components. This Software architecture, shown in Figure 1, consists of following main components:

ScaleAgent: Including all the information of an agent, such as configuration, world model, hearing messages and other functional components.

ScaleTool: Implementing the replaceable tools by reducing coupling and abstract interface, such as communication tools, path-planning tools and knowledge based tools of all kinds of agents.

ToolKit: Auto-registering, managing and all-updating tools.

ScaleCommand: Wrapped command

JobBasedAgent: Producing a task list and choosing a job to do in every cycle.
ScaleJob: Making decision and doing a specified job.

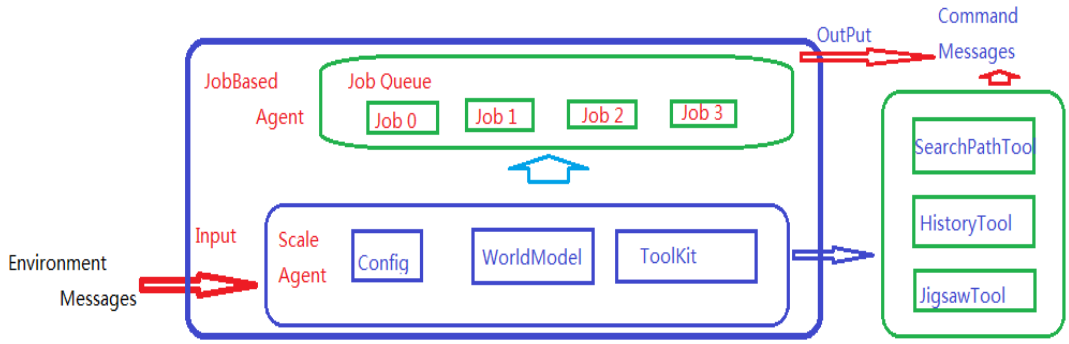

Figure.1 Software Architecture

3.Roads classification and Path planning-[1]

As there are usually so many roads in one map, it's hard for us tofind ways to go to some places or to find where the target is. To solve this question, we reduce the number of "roads" by classifying them according to their position. We divide them into three different kinds. One is called the entrance, for this kind of roads are connected to the buildings, so if you'd like to go into the buildings you must pass these roads, the second is called avenue, for they can be combined into a long street, the rest is cross, these roads are connected to more than three roads.

After the classification, we make a connection of them to form Jigsaw Avenues, and Jigsaw Crosses. For different targets (like the buildings we may want to observe or extinguish, the roads that need cleaning), we firstly find out where they are, and then we add them to their neighbor Jigsaw Avenues or Jigsaw Crosses (If the spot is connected to two Jigsaws, it's Ok, because we may choose the one which has a smaller Jigsaw number). Finally, here come the Objective Jigsaw Avenues or the Objective Jigsaw Crosses. As there may be many jobs added to one Objective Jigsaw Avenue or one Objective Jigsaw Cross, it will be easy for our agents to finish all the jobs along one avenue or cross. It can save more time as we just need to discuss who is fitter for one Objective Jigsaw Avenue or one Objective Jigsaw Cross once and for all. On the other hand, after we finish all the jobs along those roads, we may let the agents check if there are other emergencies, like the other roads along the avenue being blocked or other buildings catching fire and what we didn't notice before. So these agents will still keep observing the avenue or cross to make sure there aren't more emergencies!

The whole structure is like the flowchart below:

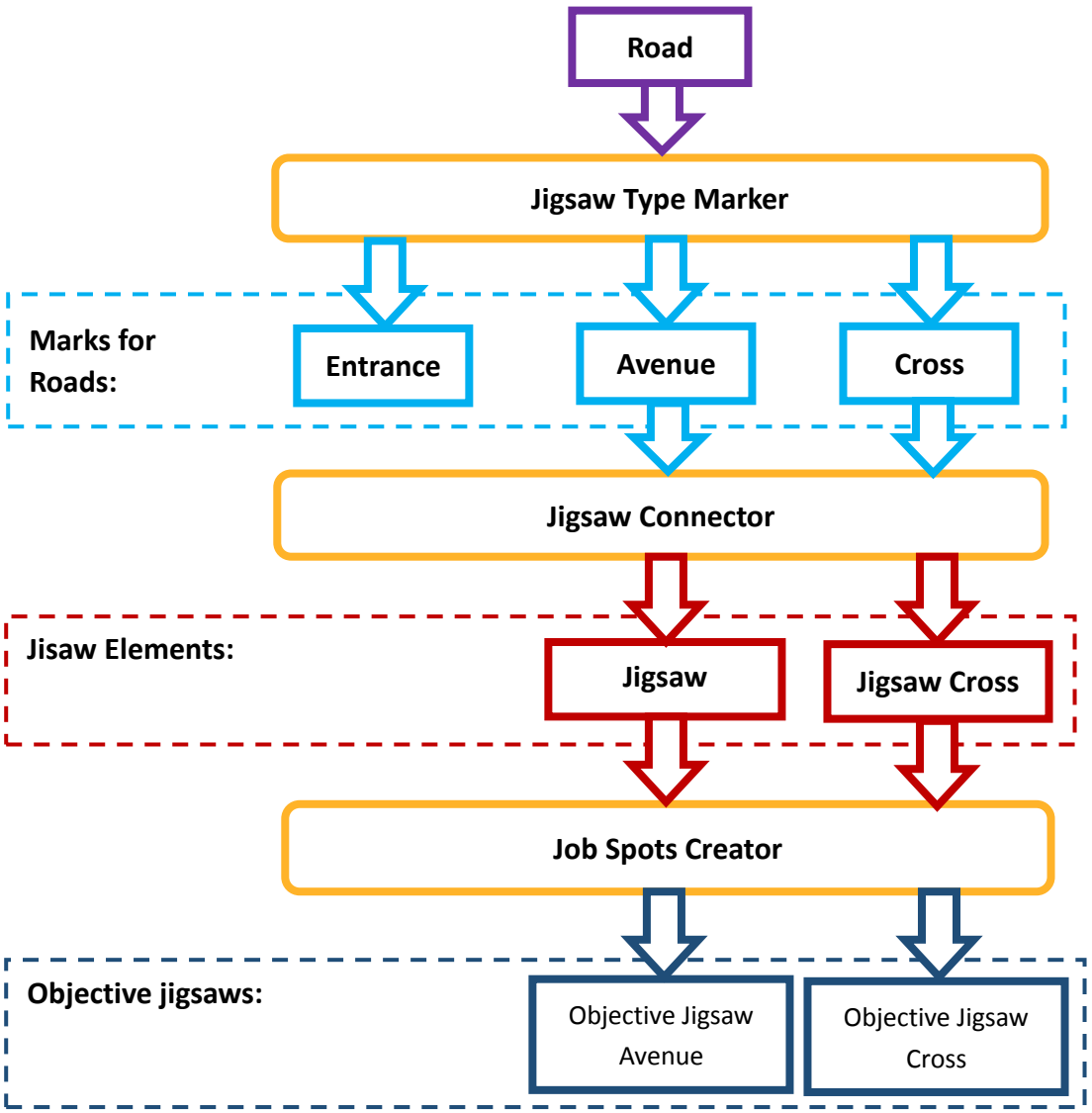

Figure.2 Jigsaw structure for road classification

These are viewers for our division strategy. Crosses and avenues are correctly recognized, which can be used to speed up our path planning algorithm. We use A* algorithm-[2] to plan path. And the nodes are recognized crosses and avenues. Fig 3 shows an example of road classification. Dark yellow parts are avenues, and light yellow parts are crosses.

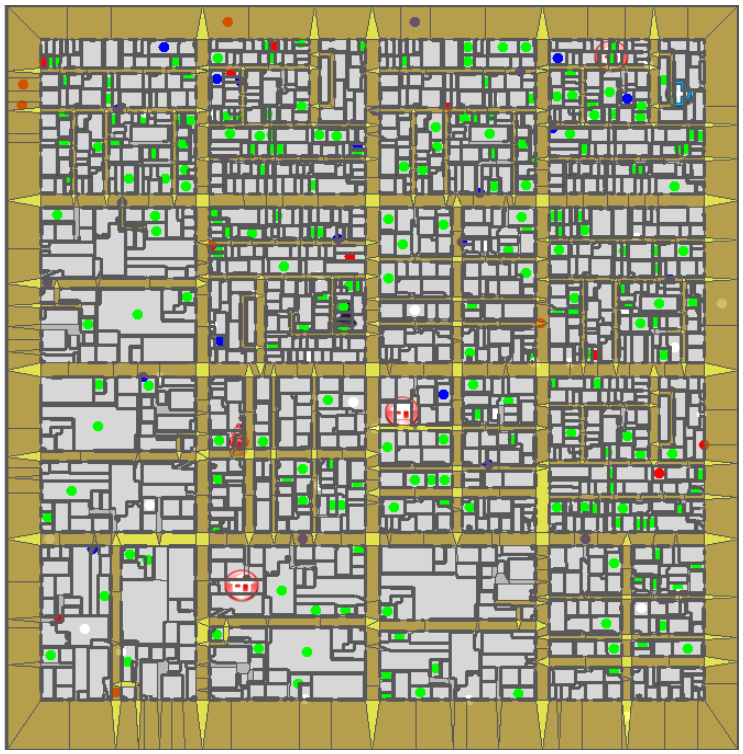

Fig.3 Example of classification

4. Partition Algorithm

In order to make agents search map and finish jobs more efficiently, we divide the map into partitions. First, we figure out the regions surrounded by the shortest paths, which we name as “District”, as showed bellow.

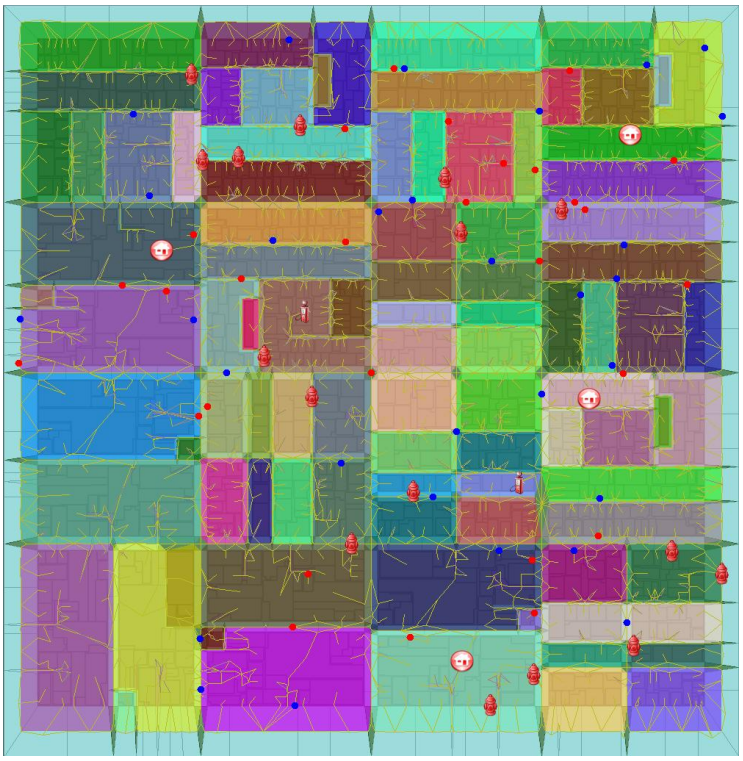

Fig.4 Example of District(VC2)

Then, we classify the districts by using k-mean algorithm, into partitions. To use these partitions, we generally create partitions based on the number of specific agents.

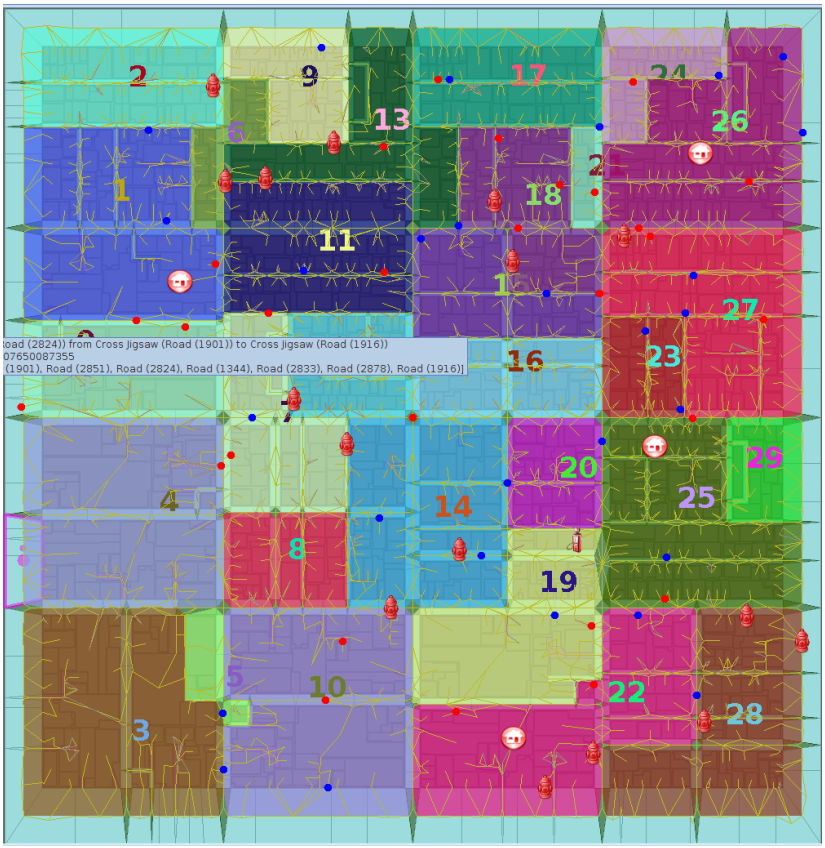

Fig.5 Example of partitions(VC2)

Here is the detail.

a. Find the simple circles based on ScaleJigsaw. By traversing jigsaws calculated by CreateJigsaw clockwise(or anticlockwise), we get circles that have only buildings in it(no roads). Every time we reach a crossing, we choose the avenue whose angle(0~2pi) to the original avenue is the smallest.

b. Next we calculate the districts. Districts is the roads in order surrounding a circle mentioned above. We traverse the avenue and cross list attained from step 1, in the meantime making it in order. 3. Putting all the buildings in different districts. A building belongs to a district if and only if the center of the building is within the polygon representing the district.

c. Clustering districts into partitions using K-Meam algorithm.

5. Partition Assign Algorithm

After our partitions have been established, how can we assign each partition to each agent?

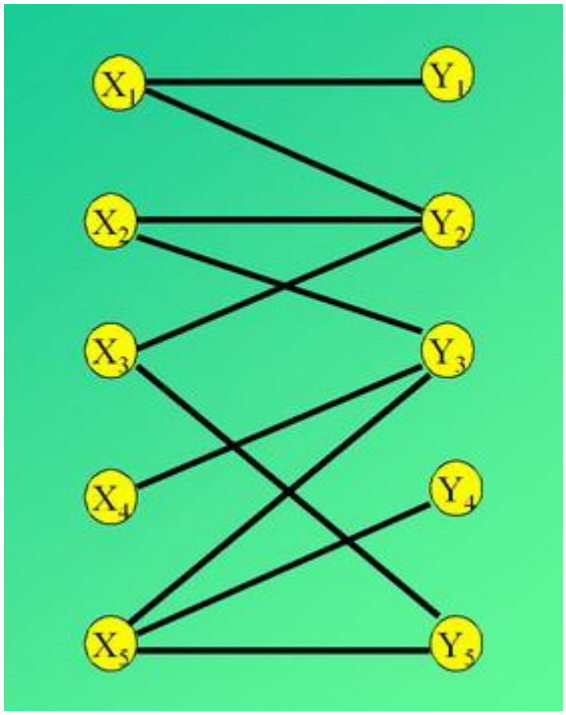

Fig.6 bipartite graphs

Suppose our agents and partitions are some “nodes”, and there are many “edges” between each of them, each “edge” has a “weight”. For our problem, the weight is the distance of agent and his partition. So we can represent this problem by the figure above. In this figure, we use Xi represents i th agent, Yi represents i th partition, and the edge between Xi and Yi representes the distance of them. Our goal is to assign the Xi to a nearer Yi.

In the case, we choose the Hungarian algorithm. And there are some results below:

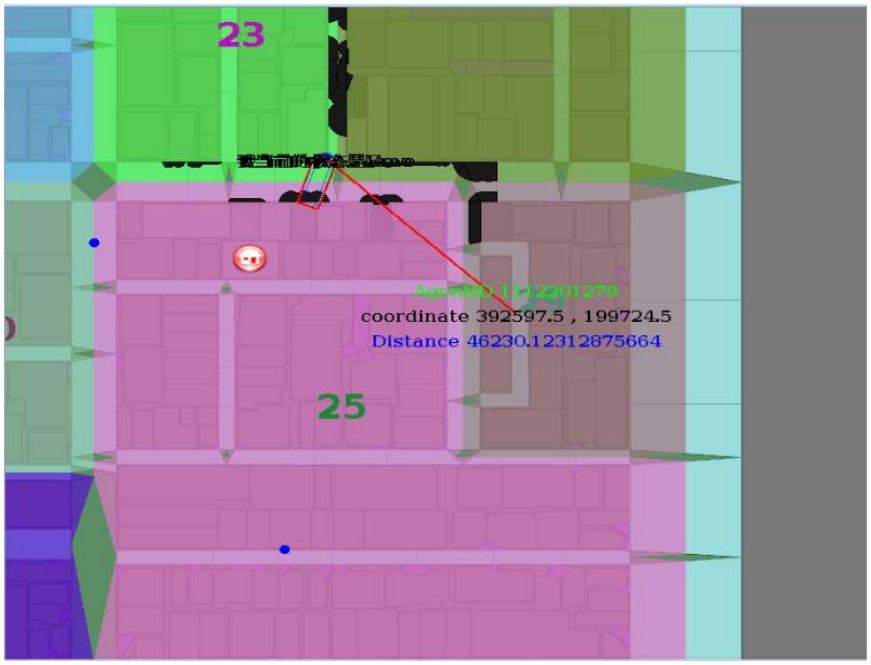

Fig 7.Partition assign

In this figure, the red line is point to the 24th partition’s center.

This agent is assign to the 24thpartition, it’s a near partition of him, right? But it can only confirm each one have a “nearer” partition, not the “nearest” one.

6.Communication model

Communication is vital in our overall architecture, however, communication condition varies by maps. When our agents build a self-adaptive and generate strategy, they must have known the real world. Some messages are vital for them. Therefore, we must ensure that these messages must pass to them in time,so, for us, we must know every channel's actual receiving rate detection and dynamic channel subscription and optimized message-sending method. The following are the details to implement our goal.

Channel condition detect

In the first cycle when agents' commands are no longer ignored, agents begin to sub-scribe different channels and send detecting messages.

In the second cycle, agents receive messages from the subscribed channels and analyze each subscribed channel's receiving rate. To make this information fully shared, every agent will send the condition messages to every channel in the same way that detecting messages are sent.

In the third cycle, agents collect condition messages and the total receiving rate can be calculated from these messages. Thus agents will have a real bandwidth for every channel which will be used in the channel assigning process.

Channel assign

Before we assign the Channel, We have done a lot of experiment to obtain every agent's needs for message transportation in our strategy system, and then we calculate the Channel ratio of every kind of agent. After that, we assign the channel to the agent according to the real width of the channel and each kind of agent's needs.

Message type

We divide our message into two parts.One is message which can be sent through both radio and voice channel. The other is voice only which can only be sent through voice channel. Our message architecture is based on task allocation, and our task allocation consists of many task points. So, our message type is based on the task points.

First of all, we define some kinds of task points as we need to be consistent with the agent's task assignment.

Secondly, we divide these points into several parts for every agent. Then, we have designed some suitable message for every kind of agent. Design the priority empirically for every message. It is useful when the radio channel is limited.

Finally, we also define some kinds of message for the general type whose receivers are not one type. The message sending is effected by the channel condition and the change of the surroundings.

Encode and Decode

Besides the sending policy that matters on the usage of channel bandwidth, the codec also plays a vital role. Every agent sends message one by one, but when we encode the message, we

combine a group of message which have the same priority and property together. To simplify the coding of messages, Java annotation is used to mark the order of message properties then we compress the message into one or two bits and send them together. When we receive the bits from messages, we must decode the message for the agent.

6.Fire Simulator and Convex Hull

Fire Simulator

In order to improve the performance of extinguishing fire, we make use of the fire simulator to predict the real file condition. Our fire simulator is based on the fire simulator of the kernel, other than some map-based arguments. Therefore our second focus is on the prediction of such arguments, which specific the spreading speed of fire, transferring of heat, and so on. Such prediction is based on the difference of the predicted map and the real world. Correction will be made according to the difference until the energy difference between actual world and the simulated world seems the same. To make the correction more accurate, this procedure is iterated more than once.

Convex Hull-[3]

The strategies of our fire brigade are quite simple but efficient, due to the use of fire simulator. There are just several jobs for FB, namely, extinguishing fire, confirming fire condition, and refilling water. Another outstanding design of our fire brigade is considering the convex hull of a set of fire buildings. As mentioned above, the fire simulator can predict the buildings on fire, so we can make a partition based on them. After separating the building on fire into different sets, we calculate the convex hull of each set. The convex hull is foundation of extinguishing. This is natural to us, since we all know that cutting off the connection of buildings on fire and those not on is a normal and efficient approach.

7. Agents

Ambulance Team

We add a new system called “BestTargetForNumber”. This system runs when ambulance team start rescuing civilian. It is designed to judge how many ambulance teams can save a civilian successfully. If an ambulance team think the number of ambulance teams is enough for saving the target successfully, it will leave the target the and go to save another target. The system avoid the situation that while one target are rescued by many ambulance teams, another target died because no ambulance team rescue it. The system can save the most civilians in the same cycle.

We add a new system called “Lock”. This system runs when a target should be ignored for some cycles, for example, the target which is out of reach. It can help ambulance teams to choose reasonable target and give up hopeless targets in next cycle.

We optimize the “RescueJob”.Previously, ambulance teams must save agent first. Ambulance team now can choose more reasonable targets according to the factors such as buriedness, time to death and so on, but need not to save the agent first. It can help ambulance team save more people.

We also made some strategies to deal with the poor channel. Our ambulance teams will search targets according the clusters in the map. It will help our ambulance team search more targets and avoid the situation that too many ambulance teams do the same thing at the same time.

Fire Brigade

The main task of fire brigade is to fight against the fire and put out all fire areas or control the spread of fire. Our strategy of fire brigades is to control the fire from the hull of fire buildings, while the fire spreading direction and the civilians' locations will be taken into consideration on the decision on which side should fire brigades begin extinguishing.

To optimize the extinguishing process, the position on which agent starts watering should be carefully chosen. For example, it is not encouraged to extinguish while standing in another building with fire. Besides, agent should start watering the building based on the knowledge of both change set and hearing, but the reliability of heard message should be reconsidered after extinguishment.

Police Force

Clear mode:

In order to improve the efficiency of PF, we want to make some changes there. In Fig 4,we plot our thought of our new clear strategy.

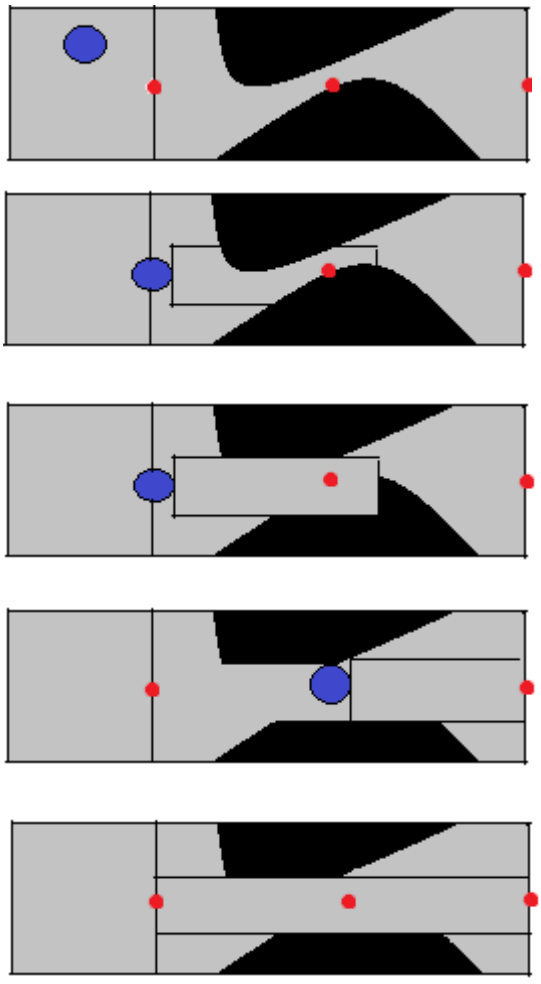

Fig.6 new clear strategy

The three red points here represent the moving model(one edge's midpoint, road's midpoint, the other edge's midpoint).Our PFAgent arrivesat the start red point, clearing a path to the road's midpoint, and then arrives at the road's midpoint, clearing the rest part of the road. As the result, we can obtain a path through the three red points.

The method above is to clear the road, if our clear object is cross or entrance, we can use a more simple command: AKClear to clear.

No communication mode:

If we detect a bad communication situation, we want our PFAgent to communicate by getting together every 15 round. For example, we can act as SOS's center-agent, choosing a special agent, and then the others get together around him.

Spot assign:

Spot is a concept like task. Our PF may use these Spots: HighwayBlocked; JigsawBlocked; SelfBlocked; TargetBlocked; ClusterOnFire;

HighwayBlocked;Here we explain the HighwayBlocked Spot in particular. Highway is a concept that explains a set of road, constructing a huge "road", it is more widely and touch to a lot of small roads. We can use highway to somewhere quickly.

Each PFAgent will find some closet Spots, here we use the Dijkstra algorithm. And to make sure everyone can move to the center-agent fast, we use MST algorithm to make sure every PFAgent is link with each other.

Spot assign problems are our new PFAgent's key points of research. We prefer to use some new AI algorithm such like Genetic Algorithms, Ant Colony Algorithm.

Convex hull for fire district

In order to make FB work easier, PF need clear the roads outside fire district. For this reason,I decide to calculate fire district's road convex hull, and make them passable. Here are the result below to show you the fire district's convex hull:

8.Acknowledgements

At last, we would like to thank all of the server developers' team for maintaining the rescue simulation server and for their technical support.Thank RoboAKUT and SOS and other leading team for their source codes which inspire us a lot. Thank Yang Genmao, Feng Huan and Jin Yifan for their selfless help in many aspects.

9.References

1. RoboCup TDP ZJUBase 2013
2. A* search – Hart,P.E.,Nilsson,N.J.,Raphael,B."Correctionto'AFormalBasisfortheHeuristic DeterminationofMinimumCost Paths'".SIGARTNewsletter, 1972,37:28–29
3. Jarvis march for convex hull– Cormen,ThomasH.,Leiserson,CharlesE.,Rivest,RonaldL.,Stein,Clifford.Introduc-tiontoAlg

orithms. MITPressandMcGraw-Hill,2001