Robocup 2011– Rescue Simulation League
Team Description Paper

Brave Circles (Iran)

Mostafa Movahedi, Mahdi Shahsavari, Abbas Abdolmaleki, Sajjad Salehi, Hamed Shahbazi

Sheikh Bahaee University
Isfahan-Iran
http://robotic.shbu.ac.ir

Abstract. RoboCup Rescue simulation competition is not only a competition but also an infrastructure and a benchmark to test different theories in computer science. Brave Circles 2011 try to introduce some of the basic ideas in multi agent intelligent systems to have a good estimation of disaster space and to have an appropriate action in this environment. This is a hierarchical disaster management structure as an inspired model of a real space disaster management system. In this structure different methods based on artificial intelligence and computational geometry employed to solve some classic problems of rescue simulation. BraveCircles 2011 by using some of mentioned methods achieved 2nd place in AUTCup2010 and 3rd of Khwarizmi Robotic National Competitions.

1. Introduction

In the RoboCupRescue Simulation, rescue agents such as ambulance teams and fire brigades act in large urban disasters. Soon after a large earthquake, buildings collapse, many civilians are buried in the collapsed buildings, fires are spreading, and it becomes difficult for rescue teams to pass roads because these are blocked by debris of buildings. The urban area called the disaster space is simulated by the RoboCup Rescue Simulation System (RCRSS). Heterogeneous intelligent agents such as Firebrigades, police forces, ambulances, victims, etc. conduct search and rescue activities in this virtual disaster world. The objective of rescue agents is, coherently, to minimize damage resulting from disasters. To do so there are some classic problems like path planning and task allocation that should be solved. In this TDP the strategies and methods which are employed by BraveCircles2011 for disaster management and solving mentioned problems are explained.

2. Fire Brigade

Fire brigades are responsible for two general tasks:

1. Attempting to reduce the ignited area
2. searching for buried and damaged civilians and informing the Ambulance team

2-1. Priority Extraction Using Delayed Rewards in Fire Brigade Agents

Action priority extraction still plays a powerful role in decision making in multi-agent systems. Using direct rewards are usually not as easy as delayed rewards; therefore a method of priority extraction using delayed rewards is described here.

We have used a delayed award approach and machine learning algorithms for creating a decision tree. With the aim of the decision tree we gain a constant extinguishing strategy using the extracted priorities.

2-2. Problem and Method Description

It is very important for fire brigades to find sequence of fired buildings to extinguish which maximizes the fitness and prevents the fire from growing more.

The method has five steps which are described here. For each action the maximum number of parameters which may be responsible for action's results is selected. Each parameter is clustered to parts by the clustering algorithm. The total discrete space is calculated and a subspace is selected for testing. After testing the test space and recording the delayed rewards the learning tree is used. If any of the parameters are removed in the learning tree results, the tree in learned again until there is no removed parameters

2-3. Clustering Phase

Each action's parameters are clustered like the way introduced in [3].The parameters are divided to more clusters as long as there is a large difference between two neighbor clusters test results. Clustering algorithms like K-means can be very useful here. After this phase, value of parameter p of action a is in one of the clusters ([1], [2],…, [n]) where n is the number of clusters.

The clustering phase changed the parameter space to a discrete space by defining some ranges for our attributes.

In the clustered model we map a range of the specific parameter to a number from N= {1, 2, 3 …} to reach discrete space (each number defines a cluster).

The attributes of the building we have used in the modeling are described below:

1. The volume of the building, called p1 which is got from the set {1, 2, 3}.
2. The kind of the building, called p2 which is got from the set {1, 2}.
3. The fieriness of the building, called p3 which is got from the set {1, 2, 3}.
4. The distance of the fired building to the fire brigade agent, called p4 which is got from the set {1, 2}.
5. The safe area of the neighborhood buildings, called p5 which is got from the set {1, 2, 3, 4}.
6. The distance of the fired building to the refuge, called p6 which is got from the set {1, 2, 3}.

The last attribute is pretty much important, since when the quantity of water of the fire brigade is finished, he should refill it.

So in this model we would have 3×2×3×2×4×3=432 different kinds of buildings.

2-4. Test Phase

We wrote a java program which saves the fitness values and stores it to a database. We tested all types of values once. This means that after that the process of simulation begins, and in the first cycle of each simulation, the fire brigades read the type of fired building which should be tested from the database and try to find and extinguish these kind of buildings first. After the simulation, the agents write the fitness of the current simulation to the database. Here the fitness is the sum of areas of all safe(less damaged) buildings.

Since we have used the fitness value at the end of simulation it is therefore dependent to many values that may not be taken into account in our test. Therefore what is actually considered here is “How Good Our Agents are Doing” [4] in overall, not the actual reward that they receiving by extinguishing each building. We maximize this fitness by the tree method and the delayed reward [5, 6].

2-5. The results of the premier simulations

The results of the simulations based on all kinds of different conditions which building could have are shown below in diagram 1.

The average of fitness is 3265.979 with the variance value of 823133.4.

The result obtained in this stage where not reasonable at all and we actually didn’t extinguish the buildings the way we should have had.

2-6. Feature Extraction:

After getting the fitness of all possible condition, the priority of fired buildings and the effect of selecting a building type first, on the output result is found. For this purpose we used a machine learning package called Weka.

Weka usually can estimate the decision with some error. Its estimations are very close to the real functions of input to output.

We used the RepTree package of Weka[7] to learn a decision tree for our purpose. The output decision tree is shown below.

```
p5 = 1
|   p1 = 1
|   |   p6 = 1 : 2966.73 (8/297801.75) [4/1125813.75]
|   |   p6 = 2
|   |   |   p2 = 1
|   |   |   |   p4 = 1 : 3332.67 (2/139129) [1/261121]
|   |   |   |   p4 = 2 : 3844.67 (3/190374.89) [0/0]
|   |   |   p2 = 2 : 2610.83 (5/247656.96) [1/39124.84]
|   |   p6 = 3
|   |   |   p3 = 1 : 3313.67 (2/74802.25) [2/1499767.25]
|   |   |   p3 = 2 : 2859.11 (2/48841) [2/269125]
|   |   |   p3 = 3 : 4272.56 (2/28392.25) [2/1790716.25]
|   p1 = 2 : 3361.16 (27/708741.48) [9/457420.67]
|   p1 = 3 : 3208.78 (26/405808.69) [10/533876.76]
p5 = 2
|   p4 = 1
|   |   p3 = 1
|   |   |   p1 = 1 : 3186.33 (5/312210.8) [1/193600]
|   |   |   p1 = 2 : 3630.33 (2/796556.25) [4/496744.25]
|   |   |   p1 = 3 : 2668.13 (3/229016.67) [3/383240.33]
|   |   p3 = 2 : 3629.51 (10/1495770.44) [8/733544.38]
|   |   p3 = 3 : 2979.95 (11/751931.09) [7/723592.29]
|   p4 = 2
|   |   p1 = 1
|   |   |   p2 = 1
|   |   |   |   p6 = 1 : 2508 (3/13312.67) [0/0]
|   |   |   |   p6 = 2 : 1949.67 (2/98596) [1/211600]
|   |   |   |   p6 = 3 : 2756 (1/0) [2/2043208]
|   |   |   p2 = 2 : 3209.14 (6/2425548.25) [3/1551812.92]
|   |   p1 = 2 : 3328.83 (8/802906.36) [10/619160.77]
|   |   p1 = 3 : 2899.95 (14/171686.35) [4/58913.3]
p5 = 3
|   p1 = 1 : 3530.93 (26/1236228.54) [10/1433939.19]
|   p1 = 2
|   |   p3 = 1 : 2926.39 (10/424808.44) [2/66227.86]
|   |   p3 = 2 : 2660.14 (7/660095.35) [5/635992.25]
|   |   p3 = 3 : 3289.28 (9/147031.56) [3/1205716.67]
|   p1 = 3 : 3365.56 (27/929866.77) [9/914660.02]
p5 = 4
|   p6 = 1
|   |   p1 = 1 : 3295.42 (8/302022.75) [4/698670.5]
|   |   p1 = 2
|   |   |   p3 = 1 : 4759 (4/3898838.5) [0/0]
|   |   |   p3 = 2 : 3019.75 (3/1789753.56) [1/47378.78]
|   |   |   p3 = 3 : 2754.67 (2/0) [2/414924.5]
|   |   p1 = 3
|   |   |   p3 = 1 : 3382.5 (3/391992.67) [1/1136356]
|   |   |   p3 = 2 : 3668 (2/609961) [2/602660]
|   |   |   p3 = 3 : 4979.75 (3/319176.22) [1/1743280.11]
|   p6 = 2 : 3186.47 (19/565618.35) [17/1185831.87]
|   p6 = 3
|   |   p1 = 1
|   |   |   p2 = 1 : 3955.96 (4/57954.25) [2/97581.25]
|   |   |   p2 = 2
|   |   |   |   p4 = 1 : 3599 (3/214260.67) [0/0]
|   |   |   |   p4 = 2 : 2567.33 (2/367842.25) [1/1872792.25]
|   |   p1 = 2
|   |   |   p3 = 1 : 3344.25 (3/17050.89) [1/653402.78]
|   |   |   p3 = 2 : 4136.67 (2/24180.25) [2/464881.25]
|   |   |   p3 = 3 : 2972.63 (1/0) [3/10311291]
|   |   p1 = 3 : 2825.76 (8/133132.36) [4/206548.64]
```

2-7. Decision tree Learned by Weka

By sorting the set of the fitness values in tree, priorities are determined. This means that the highest value in tree has the highest priority and the second highest has the second highest priority (in the shown learn tree (p5=4, p6=1, p1=3, p3=3) has the

highest priority). This way, a constant strategy to extinguish the fire buildings is extracted.

2-8. The result of final simulations:

By means of our constant extinguishing strategy we performed 32 simulations on the same situations of the previous simulations and we got wonderful results from that. The average value of the fitness increases to 7320.061 with variance value of 269475.2 . Diagrams show the results.

As you see the results of simulations (diagram 2) got very better and the extinguishing job of the fire brigade agent has been improved. So we can conclude that our consist extinguishing strategy is pretty much more reasonable than the old one.

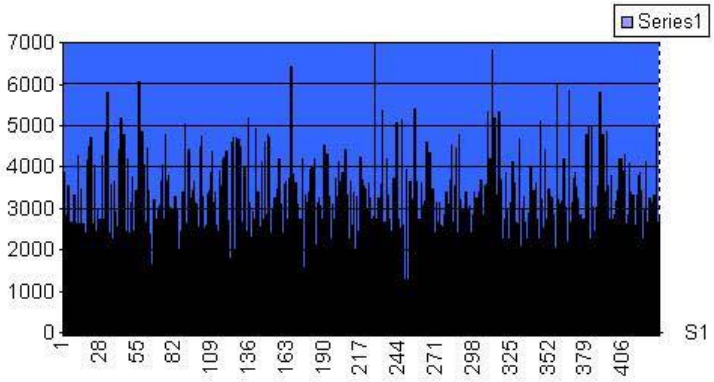

Diagram1- Simulation Results

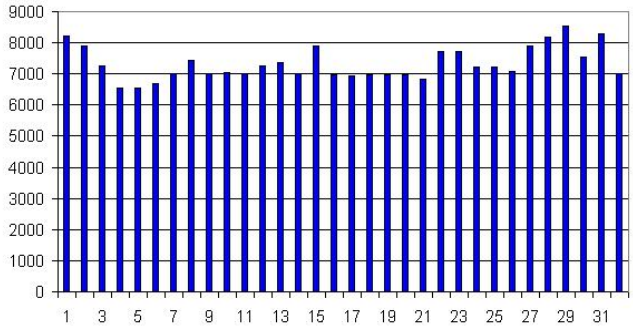

Diagram 2- Final Simulation Results

3. Agent Structure

3-1. Code structure

As shown in fig.1[4] agent communications work through RCRSS Protocol Socket. RCRSS Protocol Socket is response to make connection between Agents and kernel and consists of functions that make this connection possible. Agents communicate with each other with Message Unit.

The data, which supported by this unit, consists of two kinds. One is Data Base that is used for world model integration and the other is service, which consists of a set of missions that comes from upper layer of agent.

The exploration of other agents is as follows.

3-2. Decision making

This unit is responsible for general decisions such as Task allocation, for example, this unit will be responsible for allocating ATs to give appropriate task in Task Pivotal method.

3-3. Process engine

Missions that are given by decision making unit, reach to this unit. This unit divides these missions to some submissions which are important for finishing the mission also execution monitor supervises its runningstep by step.

3-4. World model interface

It includes a set of functions which are used and applied to update world model. If we don’t have center, commander will be implemented in one of agent’s group. Commander also sends a message in form of service both for itself and other remained agents.

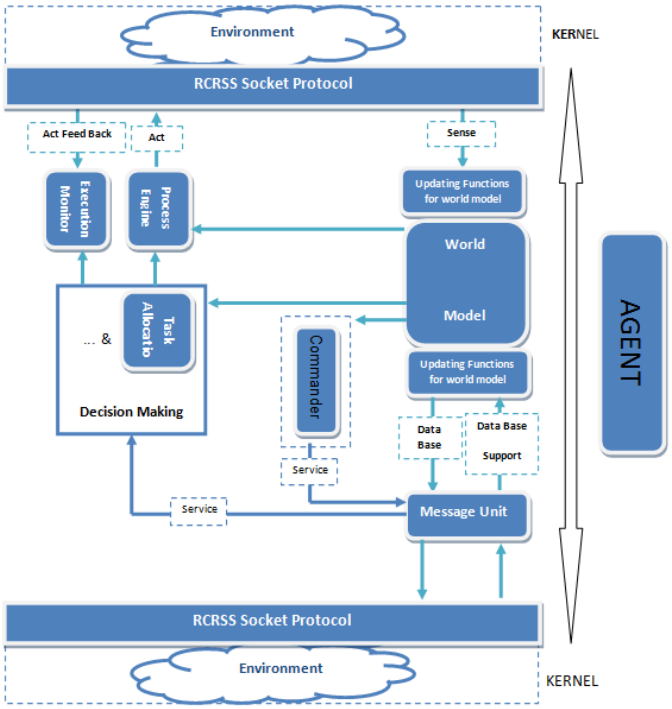

Fig.1Agent Structure of BraveCircles

4. Path planning

The path planning strategy of BraveCircles 2011 is based on modified A* algorithm for rescue simulation environment. We make a graph of map as follow. The center of each passable edge of a road is a vertex. For each road a complete graph will be generated. The final graph is made of combination of graphs of roads.

For each edge of graph if is not passable it will remove from graph. The question here is that which edge of graph is passable or not!

We use this algorithm to determine a specific path in a road is passable or not:

```
Function isPassable(road, blocks[], startEdge, endEdge)
   newRoad← subtract(road, blocks[])
   unPassableEdges[] ←unPassableEdges(road)
   walls[] ←findWalls(unPassableEdges[])
   line1←findMinBoundrySpace(wall[1])
   if(intersect(line1,startEdge) and intersect(line1,endEdge) andnot
   intersect(line1,walls[2:n]))
        return true
   returnfslse
```

In this algorithm we have some key word:

newRoad: is a shape that comes from subtracting blocks from road.

unPassableEdges: the edges of the road that agent cannot pass thru of them.

Wall: collection of connected unPassableEdges.

minBoundrySpace: is a parallel line with a wall. The distance between a wall and minBoundrySpace is greater than an agent diameter.

The left figure is a road with some block. As mentioned above we make a complete graph with center of passable edges and then for each edge of graph we compute if it is passable or not. Right figure shows that the path between edge1 and edge3 is passable.

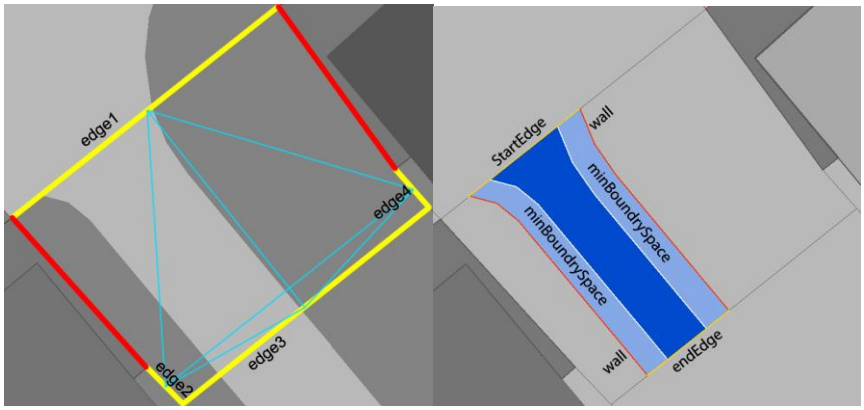

Fig.2 – Left: Area graph. Right: Minimum boundary space

5. Message manager

Agents put their messages (based on team strategy) into input of message manager. Then message manager remove redundant messages and compress remaining messages, finally find appropriate channel for each message and send them on their channels.

The schema of our message management is as below:

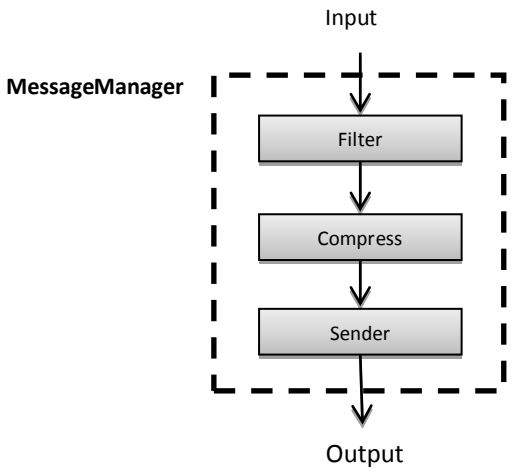

Fig.3 Message Manager

Filter: Remove redundant messages. Remove duplicate messages in this time step and previous time steps if no need to repeat. And remove messages that near agents maybe decide to send. For example if to agent saw a fiery building, one of them must send fiery building message. So this component decides that which agent must send which message.

Compress: in this layer we compress message to lowest needed bits. We use several tricks to reduce the needed bits for a message. For example in pre-computing time, we index the objects of the world. If a sample map contains 1000 building, we index IDs of buildings from 0 to 999. In this case we need only 10 bit to send the building ID instead of 32 bit!

Sender: send messages on appropriate channel.

6. References

[1] P. Stone: Layered Learning in Multi-Agent Systems, PhD Thesis, CMU 1998.

[2] H.Shahbazi, B.Shahgholi, H.Mohammadi, E.MollaAhmadi, R.Zafarani:" A Statistical Estimation of Extinguishing Utility Factor in Robotic Rescue Agents", In proceedings of the SSC5 conference in Isfahan, Iran. September 2005, Iranian Statistical Society Press.

[3] H .Shahbazi,., R. Zafarani,. 2006." Priority Interaction Using Delayed Rewards in Multi Agents Systems: A Case Study in RoboCup". In Proceedings CSICC06,Iran,571-574.

[4] Brave Circles RoboCup Rescue Simulation Team Description 2008