RobCupRescue 2013-Rescue Simulation League Team Description

<RGB (Iran)>

Arya Hadi[1], Ali Safari[1], Pedram Taheri[1], Ali Nikkhah[1], Aryaz Eghbali[2], Navid SalehNamadi[2]

{arya.hadi97, alisafar1212, pdrm.taheri, nikkhah98, aryaz.egh, navid.saleh.n} @gmail.com

[1] Robotics Research Group, Allameh Helli4 High School of Tehran, Tehran, Iran
National Organization for Development of Exceptional Talents (NODET)

[2] Electrical and Computer Engineering Faculty, University of Tehran, Tehran, Iran

Abstract. RGB Rescue Simulation Team began its activity from summer, 2012. We are using S.O.S Base Code as our base. In this paper we describe RGB Rescue Simulation Team's algorithms for finding some solutions to solve the problems concerning Rescue Agent Simulation. Here we present some new ideas and approaches regarding RCRS.

1. Introduction:

Rescue Agent Simulation is one of the computer simulation activities which is of greatest significance at emergency situations because it can help us to manage our agents and equipments in disastrous conditions such as earthquakes or other natural phenomena to have the least possible damage in the areas under our supervision.

2. Agents:

2.1 Ambulance Team Agent:

Our Ambulance Team has changed a lot comparing to our IranOpen2013 competitions Ambulance. Now we have a center-based decision making system capable of working in every map with appropriate communication. As you know, some maps do not have Center Agents but their messaging system is not that much limited therefore we chose a Captain from the ordinary Ambulance Agents and then create a Center Activity on the Captain so that the Captain can act as an Ambulance Center Agent.

2.1.1 Decision Making: To comply with the program, we would first specify the humans which are to be rescued and then assign free Ambulance Teams to rescue them using Hungarian Assignment method.

2.1.2 Specifying Humans: We give each human a score (Formula. 1) based on which we can sort them to have a list of priority for rescue operations. This list can serve us as the basis for our rescue performance.

2.1.3 Finding free Ambulance Teams: There are two known ways to do so. The first one is to use Radar Channels to send each Ambulance's state. The second one is to guess and estimate each Ambulance's state. We make use of a combination of the two methods.

$$score = \frac{1}{CTD \times buryCoef}$$

Formula. 1. CTD: Cycles to death of human, buryCoef = human's buriedness/100
We acquire CTD from our Human Estimator.

2.2 Police Force Agent:

2.3.1 Decision Making: Our Police Force is a state-based Agent with different states and different priorities for each state. We specify some agents to every state in each cycle. States are as follows.

- Agent Releasing and Refuge Opening State(ARARO)
- Fire Site Clearing State
- Crossroads Clearing State
- Civilian Releasing State
- Opening Ways, Search State
  Every Police will enter all states by their priority and skip the unimportant ones to remain in the target state.

2.3.2 Agent Releasing and Refuge Opening State:

2.3.2.1 ARARO Clustering: Clustering the Stuck Agents, Buried Agents and Refuges based on the distances between them. If we face with lack of Police Forces, then we would merge the clusters mentioned above.

2.3.2.2 Match with Euler tour algorithm: We match the clusters first with MST graph and make a close line by Euler tour from the graph (as we will make every odd neighbor points even then we do the Euler tour algorithm).

2.3.2.3 Assign agents to clusters: We assign the agents to clusters wit Hungarian.

2.3.2.4 Picking a target: In this step the Police Force will choose the nearest entity in its cluster and then it would make a reachable route to the chosen entity.

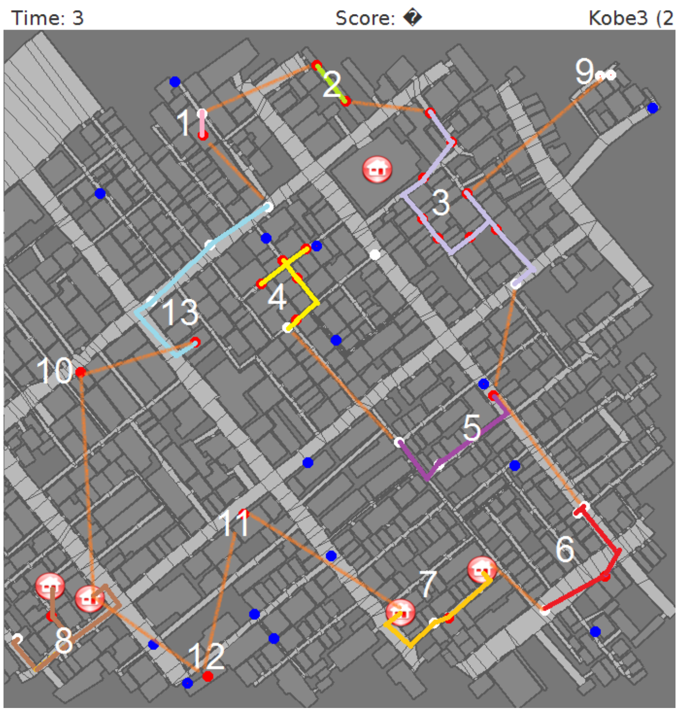

Fig. 1. The color lines are clusters and the orange lines are matching them making a MST.

At the end of this state all agents will be reachable to other agents, refuges and almost all the map.

2.3.3 Crossroads clearing state: In this state we would first detect crossroads and end of the Roads. Then we will clusters them using K-Means algorithm. After that we would match them by getting Euler tour of their MST. We assign the clusters to Police Forces by Hungarian assign method then each Police Force clears its cluster and the Path to next cluster

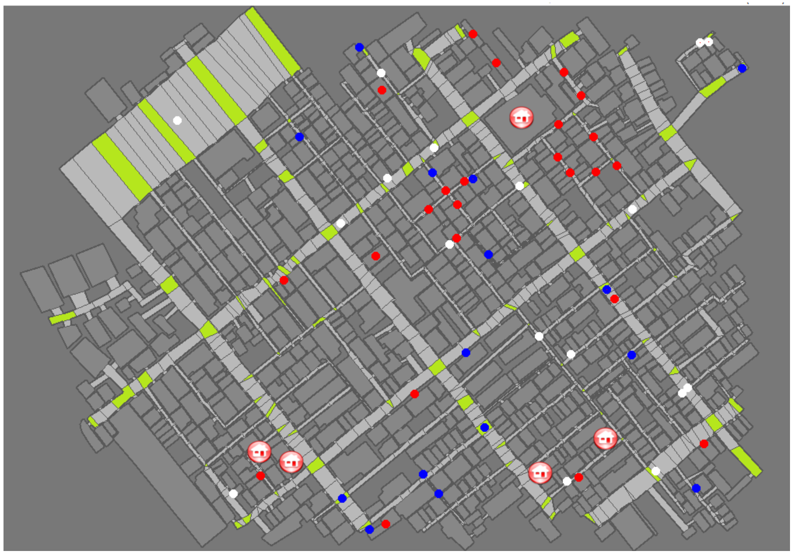

Fig. 2. The green parts are important roads.

At the end of this state all the map will be reachable as you see above.

2.3.4 Civilian clearing state: In this one for each Police force if there were a civilian in its cluster (its Search cluster) that is not cleared yet it will go and clear it. **At the end of this state all agents will be reachable to all civilians.**

2.3.5 Police Move:

2.3.5.1 Move with new clearing command: In our move we first make a path to destination and then in every cycle it will decide to clear in which direction, to do so we add some points of its path to an array of points. It will add points with this conditions:

- All of the entities points (roads excluded)
- All of the roads edge with neighbors (middle point).
- Distance of the points must be lower than average move distance.

Then it will get the sum of points with coefficient of 1/distance * C (c is a learning result). And then it will enter that point as an input for the server command to clear. It will clear minimum p% of the clear distance in the chosen direction (p is an experience and learning result, by default 75%).

2.3.5.2 Move with old clearing command: In this move we've just wanted to clear minimum blockades for saving more time, but if we do this we will face move bugs that sometimes agents can't pass some open routes, then we made our reachability checking more sensitive to reduce the server bugs.

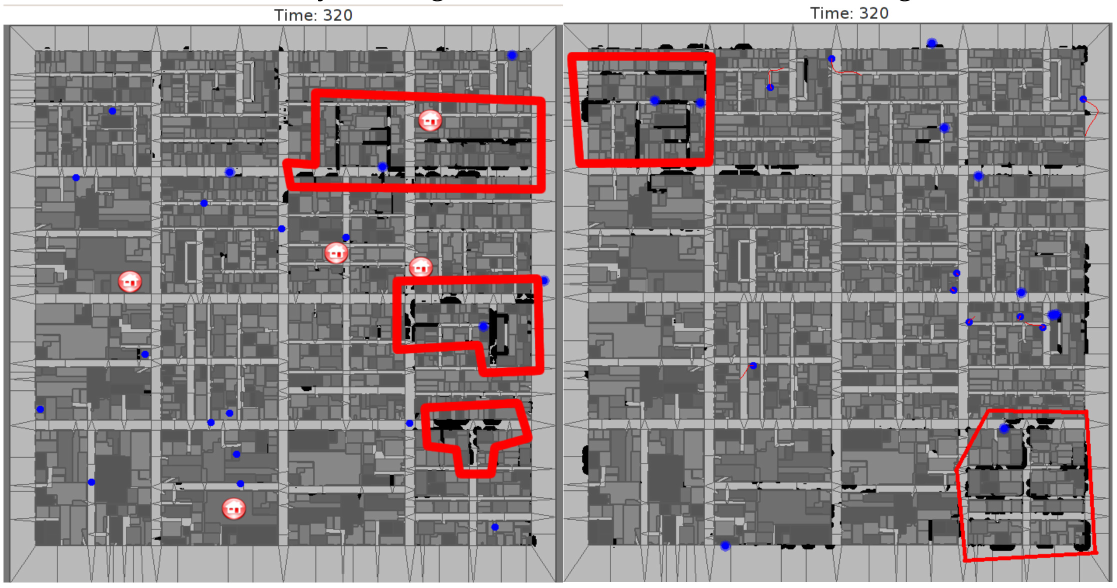

Fig. 3. Teams that clear a lot won't reach to some parts.

2.3 Fire Brigade Agent

Certainly, most important thing in Fire Brigade Agent's algorithm is, Speed, because in most scenarios Fire can burn city in just a few cycles.
We started our work with data collection about how fire can transmit in map and some other useful data. After these things, team arranged an algorithm that will explain in next section.

2.3.1 Decision Making:

2.3.1.1 Clustering: Fire clustering is the first step in our algorithm. We cluster fires using BFS on fiery buildings neighbors and find environs buildings that can transmit fire. Then each cluster will get a score depends on its size.

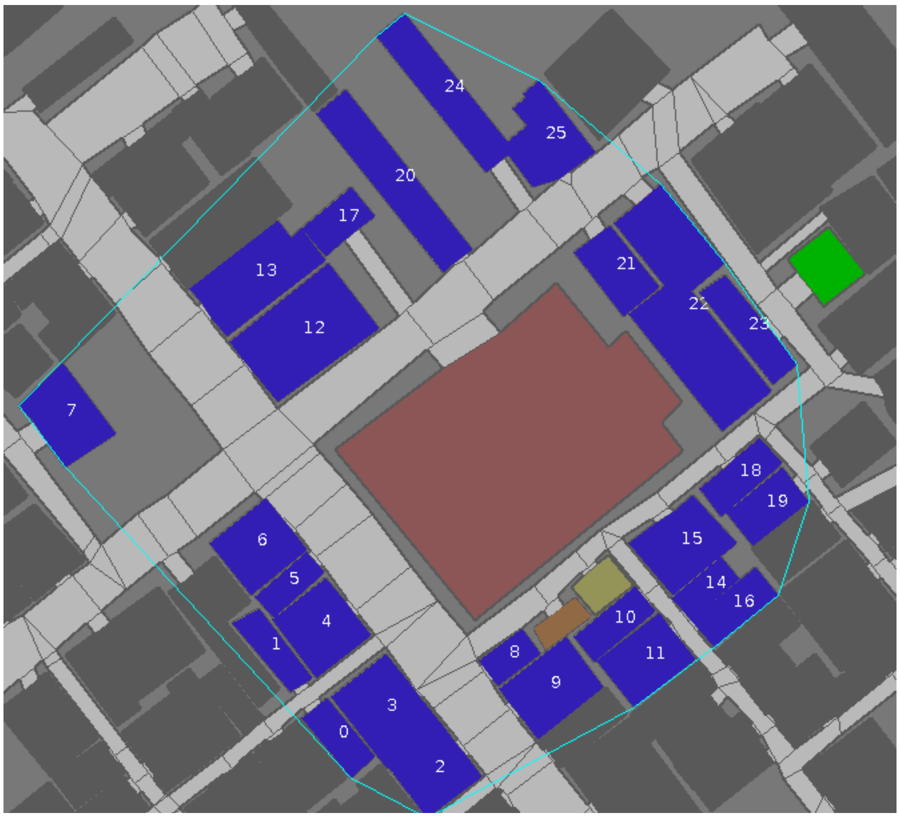

Fig. 4. The Fire Cluster shown with a Convex Hull and the purple Buildings are some buildings in around of Fire Cluster that can transmit fire to unburned buildings

2.3.1.2 Cluster Assign: In this step we will assign each agent to a cluster by their first position in map. We start agent assigning from small cluster and assign much agent as they need.

2.3.1.3 Building Assign: All environs buildings will be scored by their distance to nearest Gas Station and map's center, then each agent will assign to 3 most important buildings.

2.3.1.4 Extinguishing: This step is positioning and extinguishing using an estimator that guess how much water is need.

2.3.1.5 Search Cluster: If there wasn't any other fiery buildings, then assigned agents will search around cluster. Doubtful buildings are outer building layer of extinguished Fire Cluster.

We optimized some of coefficients using Hill Climbing algorithm and it improved this agent more than expectations. There are two examples of optimization on two customized maps:

	Before (buildings damage)	After (buildings damage)
Kobe 1	0.735	0.828
Kobe 2	0.669	0.678

2.3.2 Fire Estimator: We implemented a Fire Estimator based on data collected from Fire Simulator. Each Building has Energy and it will transfer its energy to other buildings using radiation.

Energy = temperature * building's volume

Radiation = temperature * walls overlap * Boltzmann constant / distance

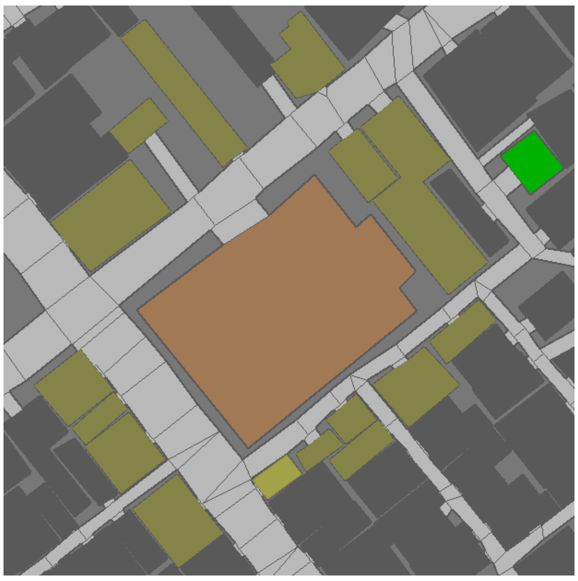
Fig.5. Estimated fire

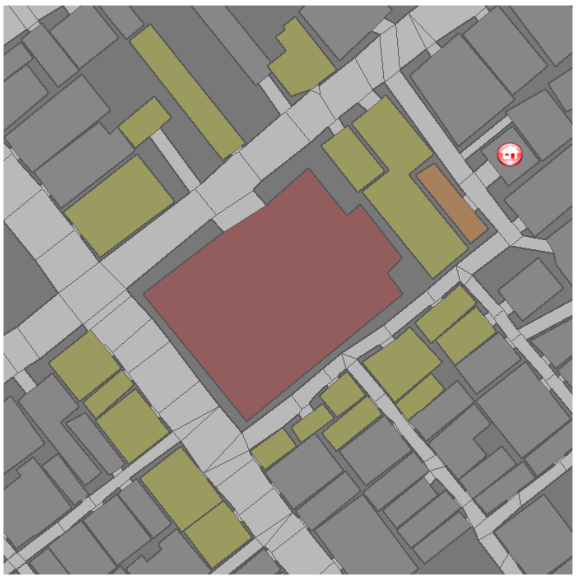
Fig.6. Real Fire

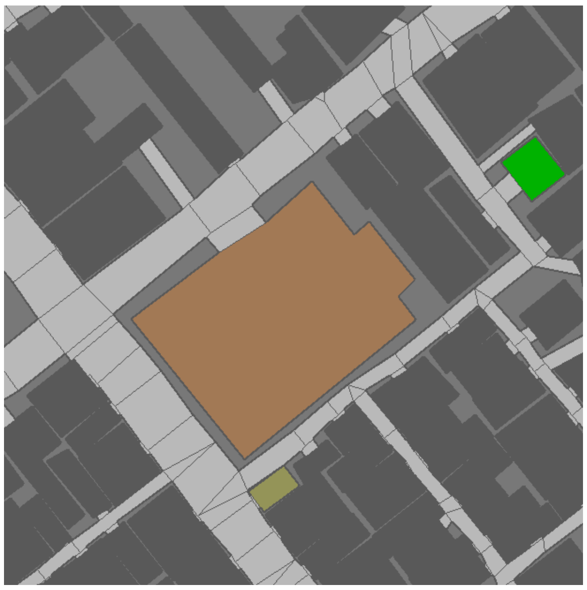
Fig.7. Last Sensed Fire

We developed a real-time optimizer for fire estimator too, to get much robust results on different situations. There are some results of optimizer on some customized maps:

(Errors are depends on variance of fiery buildings)

	Before	After
Kobe1	60% error	24% error
Kobe2	10.52% error	0% error

* Errors are calculated by variance of diagnosed fiery buildings and real server's fiery buildings

3. Search Strategy:

3.1 Civilian Search: We have a Nearest Search Strategy with lots of enhancements and improvements. First we divide the Buildings of the city into different divisions equal to the number of the agents using K-Means algorithm, and then we would assign each agent to a specific cluster using Hungarian Assignment algorithm. In the next step, every Agent is supposed to look for the nearest building in its own division to start its inspection.

3.1.1 Search Positioning: After we choose the target, we would use the center of the target as the starting point for our vision projection. After the vision projection we will have a polygon made by the area being covered by the vision lines. The intersect between the polygon and the Road in front of the target, will be taken as a position from which the searchers can view inside of the building.

3.2 Fire Search: To eliminate the possibility of having a worst case, we would have a Random Fire Search. In this part we would first divide the Roads of the city into different divisions equal to the number of the agents using K-Means algorithm, and then we would assign each agent to a specific cluster using Hungarian Assignment algorithm. Each agent would select a random Road from among reachable roads in its own division to start its inspection.

4. Reachability:

We use Dijkstra Algorithm to check area to area reachability.
In each steps of Dijkstra on edges, if the edge was blocked, we will skip it, otherwise we will iterate this task to reach the destination.
To define edges blockage we define reachable parts of roads and then if start and end points of an edge were on a non-blocked parts then the edge is counted as an open edge.

5. No Communication:

5.1 Police Force Strategy: We have created a new way to connect agents and share the information between them in No Communication maps using Voice Channel. To comply with the program, we will spot the most visited Refuges in the map. Then we would place some Police Forces in the spotted Refuges to share the information with the agents while they are visiting the spotted Refuges. Also at some specified cycles Polices change their place.

5.2 Ambulance Team Strategy: To manage Ambulances in No Communication situation Ambulance Team has a Workspace-based decision making system in No Communication situation. Workspaces are clusters of Buildings created by K-Means algorithm which can be expanded. Each Ambulance Team expands its workspace after some cycles (80 cycles). Each ambulance is given permission to work only within its own specified workspace (search, rescue, etc.). Then it gives each human a score (Formula. 1) based on which we can sort them to have a list of priority for its rescue operations based on which it will start rescuing humans. (Assigning workspaces to Ambulances is also done by Hungarian Assignment).

5.2.1 Assigning Humans to Ambulances: human with the highest score will be assigned to the ambulance with the greatest ID and if a human needed more than one agent, the free ambulances whit greatest IDs will be send to rescue the human.

6. References:

1. Hill climbing: en.wikipedia.org/wiki/Hill_climbing (last page's modification: 16 March 2013 at 23:44)
2. Optimizing Search Hill Climbing, Genetic Algorithms: cs.ucr.edu/~eamonn/205/3-optimizingsearch.ppt
3. S.O.S. Team Description Paper Proceeding of RoboCup 2012
4. The k-means algorithm: www.cs.uvm.edu/~xwu/kdd/Slides/Kmeans-ICDM06.pdf
5. k-means clustering: en.wikipedia.org/wiki/K-means_clustering
6. Minimum Spanning Tree (MST): http://en.wikipedia.org/wiki/Minimum_spanning_tree
7. Euler tour: http://en.wikipedia.org/wiki/Euler_tour
8. Hungarian Assignment: http://en.wikipedia.org/wiki/Hungarian_algorithm
9. Dijkstra Algorithm: http://en.wikipedia.org/wiki/Dijkstra’s_algorithm