RSL 2009 – Rescue Simulation League Team Description <SEU_RedSun (China)>

Guan Daqi, Meng Qingfa

School of Mechanical Engineering, Southeast University
gdq19870906@gmail.com alphameng@gmail.com
http://me.seu.edu.cn/sfzx/rescue

Abstract. After RoboCup 2008 Suzhou, China, we thoroughly reconstruct and simplify SEU_RedSun team code based on official Yab package. In this paper, we describe the code structure and some new features that have been adopted and implemented in SEU_RedSun. These new features include world modeling, communication modeling, and zone based fire controlling and civilian rescuing. Although certain features still need fully test, SEU_RedSun has gained 3rd place in RoboCup China Open 2008 based on these new features.

1. Introduction

RoboCup Rescue Simulation System (RCRSS) is a large-scale Multi-Agent System (MAS) of urban disasters. In such a dynamic, partially observable environment, the action decision making is always the primary problems which need to be effectively solved. After the RoboCup 2008 Suzhou, China, we reconstruct our code thoroughly based on the officially released Yab package.

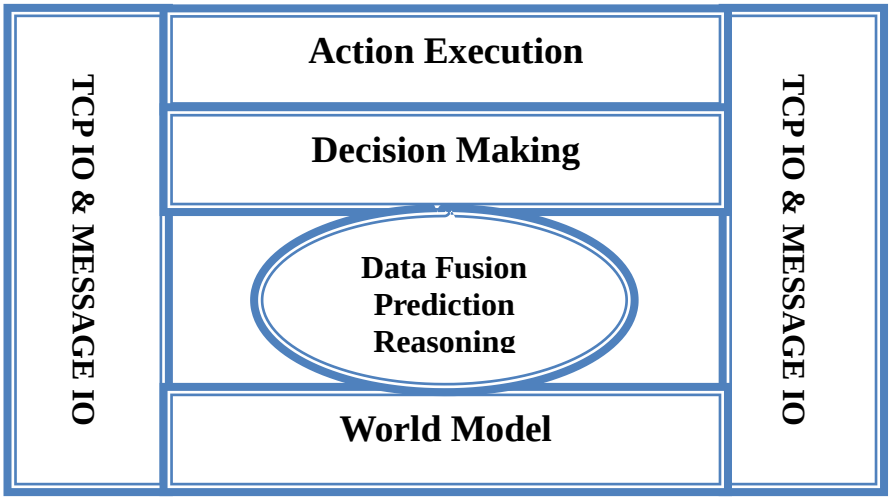

Fig.1. Code structure of SEU_RedSun

The simplified code structure is shown in Fig.1. Compared to Yab package, we create base objects for certain important objects (e.g. humanoid, building) in our world model to maintain and update their properties. We also remove some java files that we think they are just reduplicative. Anyway, we just simplify the code structures but not the decision making process.

The effectiveness of decision making needs a complete and accurate world modeling. So, we established different channel based communication models in diversified disasters for information sharing: the typical communication model and communication model under no center conditions. The latter model has some profitable characteristics such as adaptability, minimum time delay and virtually equally distributed channels. These characteristics especially enable us to build a more realistic world model under certain sharp conditions. As for decision making, both centralized and distributed approaches are adopted. Basic low level action of moving is addressed to fulfill different needs of our agents in such a dynamic and uncertain system.

Ambulances adopt totally centralized decision making approach which is treated as a dynamic assignment procedure: how to assign limited civilians to given number of ambulances in this dynamic environment. It can be very easy to understand that the total time for ambulances performing each successful rescue task is spent on two aspects: the time on road and the time on rescuing civilians. Besides, if we could estimate when civilians will be dead (called death time), we will know the maximum ambulance needs for each civilians. Thus the assignment would be achieved. In order to estimate civilian death time, a particle swarm optimization (PSO) method is then illustrated.

Decision making of fire brigades is completely distributed. Although fire prediction would help a lot in fire fighting, too many problems like building temperature predictions still stand in front of us. As inspired from practical forest fire fighting, a fire zone based approach is used to slow fire spreading. This is not the prediction way. Instead, fire brigades make different decisions based on estimation: first select target fire site, then select target fire zone and at last target building to extinguish.

2. Review of the RoboCup Rescue Simulation Platform

2.1 Structure

RCRSS simulates the real urban disasters on the computers. In this simulation environment, earthquakes collapse buildings and other architectures; destroy roads, railways and other public transport facilities; damage basic urban facilities such as electricity, sewage systems; interrupt communication facilities and information transmission, and many victims are under the collapsed houses; cause fires spreading quickly. In order to minimize disaster loss, a powerful rescue team is needed, which not only can carry out rescue tasks in disaster environment provided by simulation system but also rescue victims and save people's lives and property as soon as possible. The RCRSS is a real-time distributed simulation system that is built of

several modules connected through a network (Fig.2). Each module can run on different computers as an independent program. Each disaster phenomenon such as collapse of building and fire spread is simulated by a dedicated sub-simulator (e.g. fire simulator). Ambulance teams and fire brigades act as several independent agents. The geographical information system (GIS) provides initial condition of the disaster space, and the viewer visualized conditions of the disaster space. The kernel manages communications among the modules and the simulation.

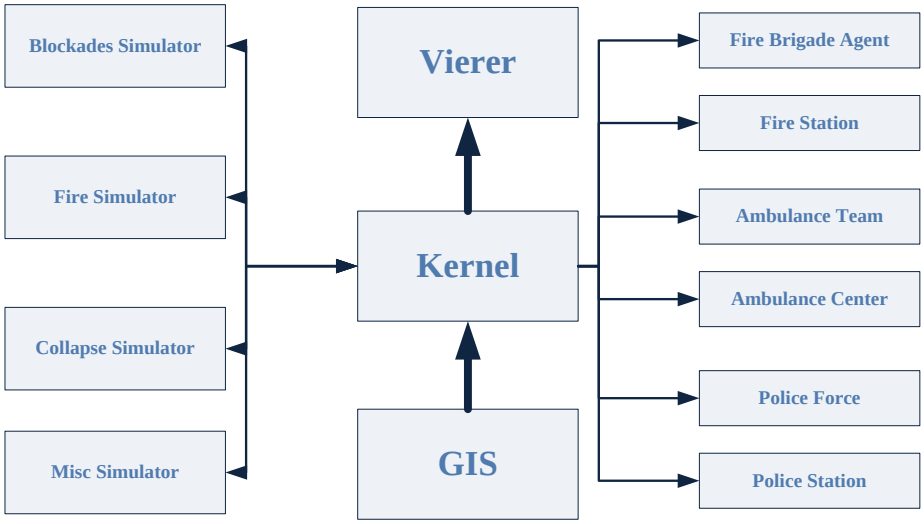

Fig.2. Structure of RCRSS

2.2 Initialization and Progress

Before starting a simulation, the kernel integrates all modules into the RCRSS as follows:

1. The kernel connects to the GIS, and the GIS provides the kernel with initial condition of the disaster space.
2. Simulators and the viewer connect to the kernel, and the kernel sends them the initial condition.
3. Agents connect to the kernel with their agent type. The kernel assigns each agent to a rescue team in the disaster space, and sends initial condition within each agent's cognition.

When all rescue teams and civilians in the disaster space have been initialized, the kernel finishes the integration and the initialization of the RCRSS. Then, the simulation starts. All simulators and the viewer have to be connected the kernel before all assignments of an agent have been finished.

The simulation proceeds by repeating the following cycle. At the first cycle of the simulation, steps 1 and 2 are skipped.

1. The kernel sends individual vision information to each agent.

2. Each agent submits an action command to the kernel individually.
3. The kernel sends action commands of agents to all simulators.
4. Simulators submit updated states of the disaster space to the kernel.
5. The kernel integrates the received states, and sends it to the viewer.
6. The kernel advances the simulation clock of the disaster space.

One cycle in the simulation corresponds to one minute in the disaster space. The kernel waits half a second for command/state submissions at steps 2 and 4, so it takes one second to simulate one cycle. It occasionally takes a few seconds according to the scale of simulation and machine specs. All agents must decide an action within half a second.

The modules of the RCRSS work as follows at the beginning of the simulation.

1st cycle: A collapse simulator simulates building collapse, and a fire simulator starts simulating fire spread.
2nd cycle: A blockade simulator simulates road blockade based on the result of the collapse simulator, and a misc simulator starts simulating humans who are buried and injured.
3rd cycles: Agents start acting.

3. World Model and Communication

Based on the disaster environment, the world model is mainly divided into three parts: simulation constants, base world objects and specific status object collections. Simulation constants would keep unchanged during a simulation. The base world objects would hold memory of objects which represent the object relationship of disaster environment. Most important of all, the specific status object collections are generally lists or sets of objects with similar attributions. For instance, a specific object collection named “buried agents” is a humanoid list that contains humanoid objects whose buriedness is greater than zero. Every cycle, the second and third parts are updated differently in that the former is updated with cognition information; while the latter is a reconstruction of world objects.

After simulation starts, agents communicate with each other to get more information of the disaster space, upon which decisions can be made properly and correctly. World model is the core memory of the disaster space for agents to predict, to reason and to make decisions. Each cycle, agents update their world model with sensed and received information. A self-adaptive, effective communication model is established to share self-sensed information as soon as possible and to build a more complete and accurate world model. In simulation environment, there are conditions where centers are collapsed. In order to share information among different types of agents under these sharp situations, we build two communication models to solve this problem. One is called typical channel communication model and the other is close loop communication model. We are going to illustrate these models bellow especially the one under no center conditions.

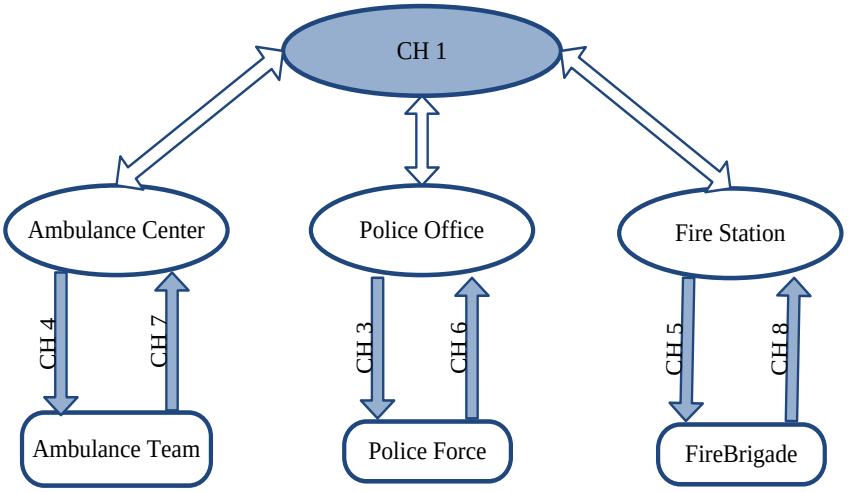

Fig.5. Typical channel communication networks

Currently, the communication of rescue simulation system is channel based with both channel numbers and channel capabilities are limited. For typical information sharing between agents, simplified model is shown in Fig.5. In this model, platoon agents only listen to their centers for message sharing while the centers use Channel 1 as public channel for message sharing. In cases where certain centers are collapsed, another communication model is adopted for information sharing, which we called close loop communication model as shown in Fig.6. We treat this model as a tree where roles of both center and platoon agents are virtually equal before the tree has been generated. After the tree generation, only agents on tree nodes act as "centers", while others on tree leaves act as "platoons" regardless their original types. These "center" agents can send and receive messages as what the centers do in the typical communication model. For the rest of "platoons", message sending is restricted to one message per cycle for the reasons of limited channel capabilities. If their messages are out of this boundary, the agents have to wait until next cycles to send them. Main advantages of this model are:

1) Minimum redundancy and time delay for message sharing;
2) Rapid message forward and traversal;
3) No communication frozen each cycle;
4) Virtually equally distributed channel capability.

In a restricted environment of limited message byte length and channel numbers, we encode messages with binary rather than string. Moreover, in order to make the information transfer more effectively, we used id-cut method to shorten the id length of certain objects and to enlarge the content of every single message. Originally, the data type of id is 32-bit integer. That is, if the agent who has received the message could uniquely determine the object with the id, the entire 32-bit integer should be added to the output stream by the sender, which consumed the communication bandwidth seriously. In fact, by classifying the objects and cutting enough bits of id to represent the corresponding object, the consuming of communication bandwidth

could decrease to an acceptable level. For example, considering the civilians, we only use the last 20 bits of civilian's 32-bit id to represent a civilian in the message. So the probability that two civilians have the same value of the last 10 bits of their ids defined as *SP*, the *SP* could be calculated as equation (1):

$$SP = 1 - \frac{C_{2^n}^{t} \cdot \left(2^{32-n}\right)^{t}}{C_{2^{32}}^{t}} \quad . \tag{1}$$

The *n* is 20 here, and the *t* is the max number of civilians (=90), so the $SP < 0.0075$. Obviously, the SP is low and the *n* (=20) is acceptable. Although this method brings some probability of communication misunderstanding, we are still content with bartering error for bandwidth at an endurable level.

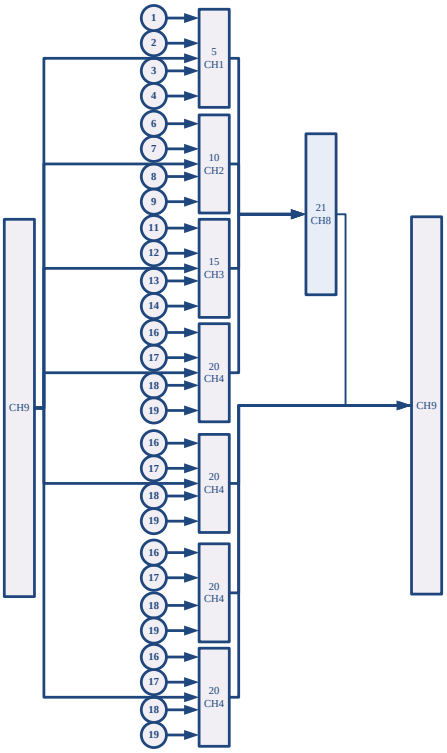

Fig.6. Close loop communication model without centers

4. Agent skills and action selection

Moving is the most essential and indispensable skill in the RCRSS based on our experiences. The effectiveness of moving action could make search and rescue

quickly and systematically. In this chapter, we will discuss in detail the approach of action selection in our team.

4.1 Moving Skill

The key of moving skill is route selection. Search and select a proper route to reach locations is the main content and the ultimate purpose of moving skill. A fast and efficient route selection algorithm enables agent to approach the target timely. In our team, the route selection is divided into two parts: try to find the fastest route to the target with blockades on roads, then, if the fastest route does not exist, find lest cost route with traffic but not blockades. In other words, by this method, we could ensure that the agent would get to the target if there is a passable route anyway, and would not stop even if there is not any passable route to target. Besides, in some special situations, we would use a non-search approach to generate the route. The three aspects of moving skill mentioned above would be discussed in detail below and an overview of our moving skill is as Fig.7.

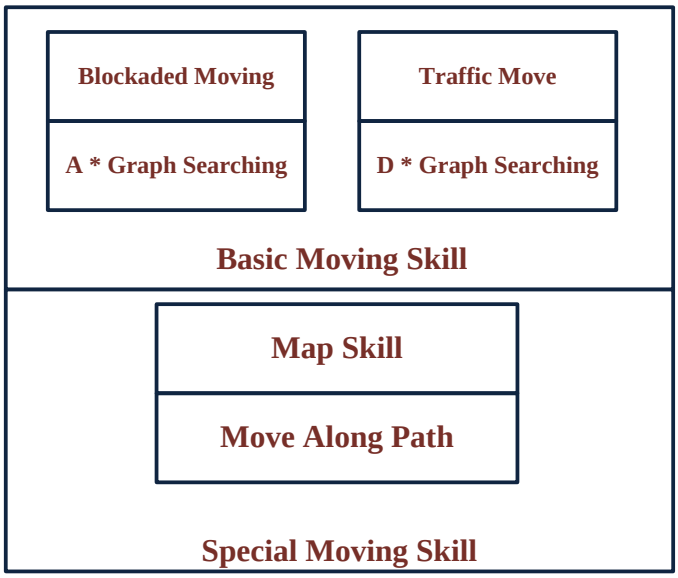

Fig.7. An Overview of Moving Skill

For the first part of route selection, A*-Graph-Search algorithm is adopted to find the result route. In order to put up the block effect of the blockades, we modify A* not to generate successors of the road which is entire blocked by blockades. The heuristic function is still the distance between the selected node and the target. Furthermore, the buildings which are not targets would not be visited. Obviously, this mechanism would accelerate the process for finding the route.

For the second part of route selection, the traffic moving skill takes both the status of traffic and blockade into account and at last generates a target route though the agent may be blocked somewhere.

Understandably, the basic moving skill discussed above is intensively depended on the information of blockades. We modified three kinds of messages to meet the need of route selection: block message, unclear message, and clear message. The block message would inform the agents which road is blocked entirely by blockades. The unclear message would hold the information that which road is passable since there are some blockades on it. The clear message is just used to inform the agents that there are no blockades on the road which is ever declared as unclear. Fig.8 describes the relationship among the three messages:

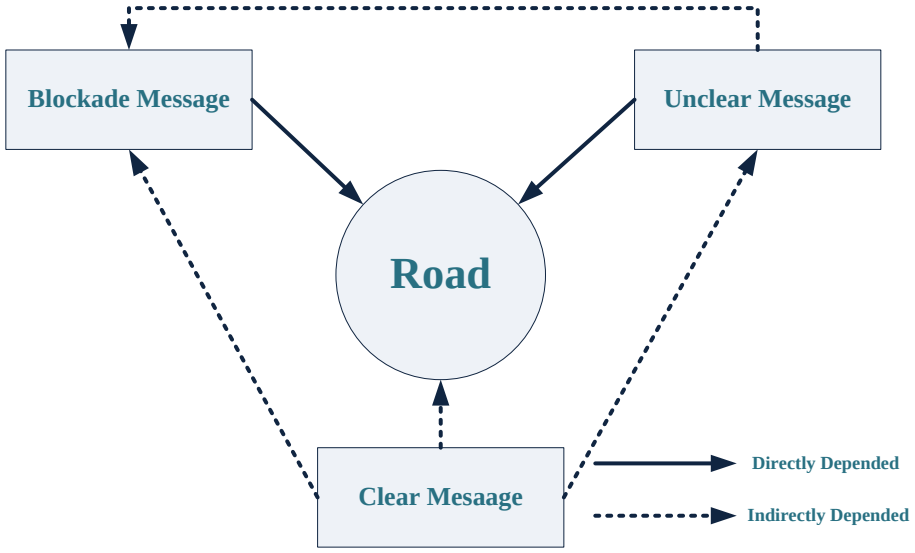

Fig. 8. Relationship among Block, Unclear and Clear Message

We also develop a particular kind of moving skill called ‘move along path’ (MAP) to deal with some special situations. Just as its name implies, the MAP makes the agent move along a specific path. Obviously, the effect of the MAP depends intensively on the method which generates paths at the beginning of a simulation. Specifically, when the agent uses the MAP skill, it could obtain as much information of the disaster as possible. For instance, search the uninformed areas could make use of this skill when there are no higher priority tasks for the agent but to search.

4.2 Fire Brigade Agent

Prediction of fire spreading is crucial for fire brigades controlling fires. If we know the fire spreading model, it could be easy for us to predict when buildings will ignite and which building will ignite at given time. Further more, we could even predict fires spreading directions. Based on the above prediction results, extinguish decisions can be made more accurately and fires will be effectively controlled. But the simulation environment supplies very limited information to fire brigades. For instance, fieryness, noted as one of the key properties of buildings, has only three

discrete values (1, 2 and 3) to represent the amount of the remaining fuels in burning buildings. Unfortunately, this important factor does not directly affect the fire spreading. Thus, it becomes very hard for us to predict fire spreading.

As enlightened from the practical forest fire fighting, we do not use the prediction way. Instead, we firstly cluster ignition buildings into different fire sites, and then cluster each fire site into different fire zones and finally control fires within fire zones. That is to say, we control fire zones to slow the fire spreading. Fig.9 illustrates the concept of fire zones.

As mentioned above, it is hard for us to establish the fire spreading model, so the decision making of fire brigades is based on estimation. Linear estimation is used for selecting target fire sites and target fire zones with highest priority. We specify some important factors of fire sites and fire zones. Then calculate a value for each fire site. After selecting fire site with highest value, the same approach is used for selecting the target fire zone in this fire site. The value for both fire site and fire zone is calculated using equation (2). Where v_i is the value of i^{th} factor, γ_i is a coefficient for v_i which presents the importance of v_i, and m shows the number of factors in different situations.

$$Value = \sum_{i=1}^{m} \gamma_i * v_i \quad . \tag{2}$$

Some important factors adopted to estimate fire sites are listed as follows.

1) Distance to fire site;
2) Number of civilians around the fire site;
3) Total burning areas of fire site.

Also, important factors to estimate fire zones are listed as follows.

1) Average burning time of fire zone;
2) Neighbor fire zones;
3) Total unburned areas.

Both the fire site and fire zone coefficients are adjusted based on observation of fire brigades extinguish performances. As for fire brigade coordination, it is an implicit way because of the distributed decision making mechanism. For instance, if tanks of fire brigades are empty when extinguishing, fire brigades have to refill tanks in refuges. The water quantity is determined by number of fire brigades in refuges at that moment:

```
In each cycle
if (FB.getWaterQuantity () <16000-FBsInRefuge*1000)
     rest ();
```

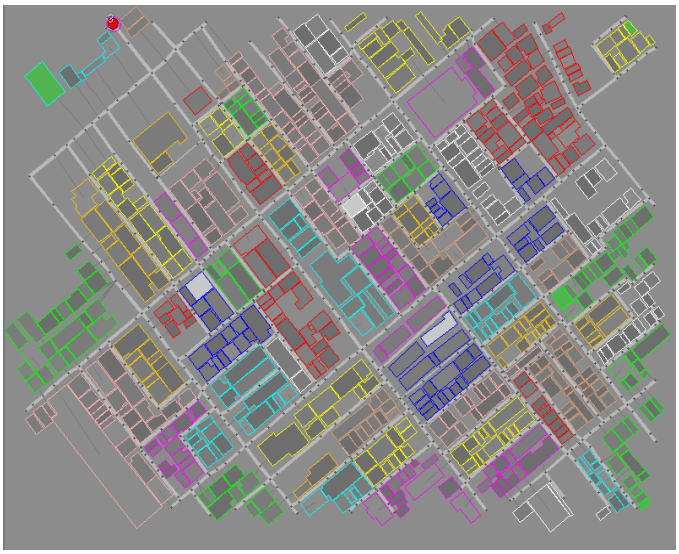

Fig.9. Fire zones with different colors

4.3 Ambulance Team Agent

As our experience, performance of ambulances could greatly affect the final scores, so the great challenge lies there. It seems proper to treat the process of rescuing civilians as a dynamic assignment problem: given *m* ambulances and *k* civilians, how to assign civilians to different ambulances to rescue as many civilians as possible?

Basically, total time for ambulances performing each successful rescue task lies in the following two aspects: time on roads and time on rescuing civilians. For the former part, time could be estimated with acceptable errors; for the latter part, time could also be estimated because the factor affects estimation most (called buriedness) is known to ambulances. But how to determine whether civilians are alive or not? So how to predict civilian death time becomes the key criterion of the problem. If we could predict civilian death time within acceptable error ranges, it would be much easier and more accurate to assign civilians to different ambulances. Because, on the one hand, we could determine whether civilians are still alive during rescue task performances; on the other hand, we could calculate maximum ambulance needs of each civilian which is helpful in our assignment. In order to solve this problem, a particle swarm optimization approach for civilian death time prediction is used.

According to the rescue simulation system, HPs of civilians are rounded to nearest 1000, while the damages are rounded to nearest 10. To eliminate the quantization errors, a group of particles are maintained and updated with sensed and shared information of different agents. Steps of PSO for estimating civilian death time are as follows.

1) Generate particles based on first observations of civilians. Each particle is a paired HP and Damage in the form of (Hp, Damage). Thus initial death time can be obtained similar to step 4.

2) Every cycle, death time declines until it is replaced by newly updated one. Update particles when new information comes from these civilians. The update model is $[\text{Hp Damage}]_{n+1} = f([\text{Hp Damage}]_n)$.
3) With this new information, illegal particles that have been simulated to current cycle will be deleted because they are just out of the quantization ranges.
4) The civilian death time is then updated based on the arithmetical average of each particle death time.
5) If no particles left, repeat step 1.

When comes to the assignment, ambulance centers take the task. As mentioned above, the total time consumed for successful rescue tasks is consist of time on roads (T1) and on civilian rescuing time (T2). If T1 outweighs T2, it would be proper to consider T1 as a more important factor when assignment is made and vice versa. So, the final assignment would be a balance between T1 and T2. That is to say, the assignment model would be α *T1 +β *T2, whereα andβ are positive coefficient for balancing. The assignment steps are:

1) Find theoretically rescued civilians (TRCs) only considering T2.
2) Assign ambulances to TRCs no more than their maximum needs.
3) With this assignment, if all TRCs are rescued, return to step 4; else adjust with a biggerβ and smallerα , return to step 2.
4) Send this assignment to each ambulance team.

5. Conclusion

In this paper, we presented a brief overview of the structures and approaches designed and implemented in SEU_RedSun after RoboCup2008. First of all, we want to build a complete and accurate world model via communications among different agents. Second, the data fusion, reasoning and prediction will be made based on this world model to support the decision making of three types of agents. Finally, various techniques have been tried or implemented in our code to deal with noisy, varied, real-time and dynamic disaster environments.

For the future, we plan to thoroughly test our code, modify minor bugs and use other Artificial Intelligence methods in order to establish an effectively cooperative team of agents in such a complex multi-agent domain to diminish the side effects of urban disasters.

6. References

1. Maziar Ahmad Sharbafi, Omid AmirGhiasvand, Saeed Ansari Ramandi Omid Aghazade, et.al: MRL 2006 Team Description
2. Stuart Russell, Peter Norvig: Artificial Intelligence: A Modern Approach. Prentice Hall, 3rd edition. (2004)
3. RoboCup-Rescue Simulation Manual, RoboCup Rescue Technical Committee (2000)

4. Meng Qingfa, Zhou Cheng, Ma Bojiang: SEU 2008 Team Description
5. T.Cormen, C.Leiserson, R.Rivest: Introduction to Algorithms. MIT Press, 1992.
6. Rong Xiong Yikun Tao Yuankai Wang et.al: ZJUBase 2007 Team Description