Impossibles RoboCup Rescue 2006 Team Description Paper

Jaffar Habibi MohammadReza Ghodsi HamidReza Vaezi
Majid Valipour Saman Aliari Nima Hazar

April 28, 2006

Abstract

Generally, Rescue Simulation environment is considered as an infrastructure in which various Computer Science (CS) algorithms are implemented and compared with each other. Nowadays, Rescue Simulation League plays a chief role in RoboCup simulation competitions. This paper presents a brief explanation of approaches, employed in IMPOSSIBLES 2006 source code. Although the basic ideas are the same as the code of IMPOSSIBLES in RoboCup 2005, we have improved and also optimized the codes in order to get higher scores. The above mentioned ideas of IMPOSSIBLES in 2005 resulted in our championship in RoboCup 2005.

1 Introduction

In this paper we are going to briefly describe our contributions to the rescue agent development and our new approaches to solve the major problems encountered by rescue agents.

In the following sections, first, common practices among agent developers will be discussed to give the unfamiliar reader the basic view point about the subject. Secondly, a good architecture for agent developing will be discussed, and the details are described in the *world model* and *transparent communication* sections.

In the fifth section, we are comparing two major strategies in developing agents *distributed* and *centralized*. Then, general approaches, i.e. employed in IMPOSIBLES in RoboCup 2005 are presented in *Agents* section. Finally, the *future works* will give the reader a clear view about what we are going to implement for Bremen 2006.

2 Common Practices Among Agent Developers

The simulation system described above is a rather complicated system. Therefore it is practically impossible to find a completely optimized solution for the

system. It means that there is no winning strategy and no one can claim to have written the best possible agents.

These limitations have led the agent developers to use artificial intelligence techniques especially related to machine learning. Of the most popular techniques for agent development are:

- Genetic Algorithms
- Neural Networks
- Fuzzy Logic

Unfortunately the majority of agent developers who use the above techniques tend to implement a centralized set of agents. That is because centers have more information than all other agents in the system. More importantly in implementing above algorithms developers usually misuse communication messages to transfer technique dependent data, forgetting that this valuable resource is only useful for, and should only be used for transferring information sensed in the virtual world.

In what follows, we are going to discuss the best practices we used in our experience of developing the award winning set of agents for the RoboCup Rescue Simulation.

3 A Well Structured Software Base

Having a layered structure is a common feature of complex and portable softwares such as networking software and operating systems. Among the benefits is that the lower layers hide away the implementation details usually dependant on the hardware.

A good software base should provide good services for the upper layers. It means that

1. It should not make any restricting assumptions.
2. It should have an easy to use APO depending on the needs of the algorithms implemented in the upper layers.

As you may have noticed the above two conditions are quite contradictory. While developing a software base one should have this trade-off in mind.

4 World-Model-Action-Command Abstraction

A useful criteria for evaluating implementation methods is consistency with "the best possible agent." First of all the best possible agent exists. Because the number of agents is limited.[1] So it is possible to sort them according to

[1] To hard disk capacity, for example!

the average score they get on all possible initial conditions. The agent with maximum score is "the best possible agent."

We can assume some properties for the best possible agent. The property we focus on here is that the best possible agent's actions are only a function of the world model. So let us define the world model.

Informally, the world model is what the agents think of the current state of the simulation environment. The world model is a subset of the information processed by the simulators. Agents construct a world model at the beginning of the simulation and constantly change it based on the information they *sense*.[2]

In the RoboCup Rescue Simulation the best possible agent is the agent who can get the best score at the end of the simulation. The score is calculated according to the current state of simulation environment. So the simulation is not time-invariant. That means *in Robocup rescue simulation environment we have to include time in our world-model.*

The conclusion of this section is that there exists an agent no worse than the best possible agent whose actions are function of its world-model. Because in some specific world model one can evaluate[3] an average[4] score for all possible actions and pick the best action.[5]

5 Distributed Vs. Centralized

There are two main approaches to development of RoboCup Rescue Simulation agents. The centralized agents send all their sensed information to their centers and wait for the center to tell them what to do in the next cycle. So all the actual thinking is done in centers.

The distributed model, on the other hand, has very simple centers that act like communication relays. That is they simply aggregate and forward the sensed information of all agents to one another. any agent decides what to do independently.

Although these models seem different at first glance they are theoretically very similar. With the assumption of unlimited communication. In this case we can easily build optimal[6] distributed agents from optimal centralized agents by implementing the centers algorithm that tell the agents what to do in every agent. Then each agent runs the centers algorithm in every cycle and does the actions corresponding to herself.[7]

Here are some point that argue in favor of distributed RoboCup Rescue Simulation agents:

1. In the RoboCup Rescue Simulation it is possible that there are no centers

[2]Or receive via communication.

[3]Based on the best possible average final score

[4]On all possible random seeds.

[5]In unrealistic case of different actions with the same preference we sort them alphabetically!

[6]Optimal in the sense of the actual decisions not performance(i.e. CPU usage)

[7]We can also build centralized agents form distributed ones easily.

or that centers exist but there is no communication.[8] In that case all agents have to decide for themselves.

2. As mentioned above in the case of unlimited communication distributed model is greater than or equal to the centralized model. But since communication takes time (i.e. simulation cycles) distributed agent can react faster to events around themselves. That is the events they sense directly.[9]

3. Currently all sensed information are reliable. There is no noise or error combined with the information. But that is planed to be implemented for the future. Also from a practical point of view there is no way one can be sure that its agents are bug-free. In case of a software failure or incorrect information all centralized agents are affected; but only one distributed agent would be affected in such cases.

6 Transparent Communication

As mentioned above "the best possible agent" can decide its action based on its world-model. Thus the information communicated among agents should be applied to world-model.

Some agent developers tend to communicate processed information. Such as an agent telling other "Come with me!." It is theoretically[10] the same as communication of raw information. Because all such messages are derived from agents' sensed information.

So the purpose of communication is to carry sensed information to complete agents' world-model. It is independent of the type of the agents.[11] Moreover it is independent of each agents actions.

Therefore agent development can be decomposed into two independent tasks. Implementing action-command and Implementing communication. The latter is well defined and mechanical task. The one thing that remains is to set priorities for information.

The model we suggest is that every agent and center have a world model for every center or agent it is communicating with. Each of these world-models contains the information our agent thinks the corresponding agent has. When an agent wants to send a message to another it calculates the difference of its own world-model with the target agent's world model. Then the differences are sorted according to their priorities and best ones are sent.

Another complication is that the number of and the length of messages are limited.Also number of messages that each agent can listen is limited to

[8] Then what would be the use of centers? actually there are two types of communication in our environment. A long distance one like radio and a short distance one like natural voice. When we say no communication we actually mean no *long distance* communication.

[9] Technically centralized agents have a "one cycle delay".

[10] With the assumption of unlimited process power.

[11] In case of limited communication some priorities are necessary so that each type of agents gets the information it needs.

maximum of 4.So there most be some strategy that specifies which agent should hear what message.There is two simple strategy to overcome this problem:

- each agent will hear one message which is sent by the center specially for it.In this case we haven't use all the agents' capacity of communication but instead we have sent each agent all the information needed by that agent.
- center will send 3^{12} messages for all of the agents and every body would hear these messages.In this case we have used all available capacity of communication for each agent and also we are sure that all the information between agents are same .It should be noted that there would be some agents that won't receive their needed information due to race condition between each agents to include their valuable information into these 3 messages.

Beside this 2 simple strategy this problem is amazing problem to solve and could be interesting case to work on for 2006 competitions.

We have implemented the above two strategy but the second one has been used more often in the **Osaka 2005** competitions because of the lesser packet sent over the network and the better performance.

7 Agents

7.1 Ambulance team agent

Ambulance team agent can be considered as the most important agent in the disaster space.This importance was a good reason to motivate us working on the injured civilian selection algorithm for this agent.

We have solved the problem of selecting between available targets by changing the goal of this selection,The primary goal of the selection is maximizing number of alive civilians at the end of the simulation run.But our goal was maximizing number of act-rescues issued by ambulance teams. To achieve this, we have devised an algorithm which only assigns exact number of actually needed ambulance teams to a civilian. First we will help civilians which only need one ambulance team to survive knowing that these civilians will die without our help ,in the second level we are selecting between civilians who need 2 ambulance agent to survive,and so on.

This is completely in contrast with strategy of rescuing the most injured civilians with all available ambulances used by most of the teams.Furthermore our strategy will eliminate extra movements of ambulances and shows to be more effective in score than other ones.

An important factor for having a successful selection is being able to exactly estimate death time for each civilian only by knowing its current *hp* and *damage*.To do this we have tried to model misc simulator behavior by logging

[12] we have reserved one message for hearing uttered messages

its output for various inputs.Using this logs and *xgraph* we have deduced that hp-time curve number of actually needed ambulance teams civilians' time-hp curve can be approximated by an exponential function with a reasonable error.

$$hp(t) = A * e^{\alpha * t}. \tag{1}$$

which A and α will be calculated for each civilian due to its current *hp damage* values. and the fact that:

$$damage(t) = -\frac{\partial hp(t)}{\partial t}. \tag{2}$$

with additional assumption that initial of civilians is 10000. This estimation has worked pretty well in all cases.

7.2 Police force agent

Police Force Agents are mainly responsible for preparing condition for other agents to do their task efficiently. As it mention Police Force agents use distributed system and Police Office dose not have anything to do expect preparing information that discussed in Communication Chapter.

The first objective of Polices is to make critical point of city reachable for related agents. It contain emergency decision that we made by some heuristic functions. For this purpose we modeled the city to a simple graph G(V,E) where V contain all motionless object and E contains (i,j) where I and j are two object that are connected to each other like road and node or building and node. Now we can easily find connected sub-graph of ours using BFS algorithms in O(E). As it was mentioned we use some heuristic function to select more important regions (connected sub-graph) in order to be joint to other region. Our heuristic function consider following factors:

- Region that contain Fire without any Fire Brigade.
- Region that contain Civilian without Ambulance or vice-versa.
- Small region that contain refuge.
- Region that contain Fire brigade or Ambulance without any refuge.
- Time.
- Length of the region.

After selecting more important region a matching algorithms used to assign these region to Police Force Agents according to current position of police forces. And Polices fined best path to join this region to the region that they are in now.

Next objective is to open some high priority blockade so each agent can go to its target faster. So they divide city to each other and each Police force open the blockade of her own part according to priority of blockades.

At last they will search the city to find civilians. Although search is not especially Police force agent's task, It is mainly done by Polices because they usually do their task sooner than other agents. At last polices that have nothing to do go to the position of injured civilian to report their condition.

7.3 Fire brigade agent

FireBrigade Agents' decision making is distributed same as other agents. Co-operation is more important for them in comparison with other agents.Also fire fighting is very unpredictable and time variant.As a result it is not possible for a team to have a single fire fighting strategy during the simulation.

Impossibles team have used various kinds of planning methods from the beginning to the end of simulation. In early cycles the used strategy is called *first attack* which is responsible to extinguish as many as fire sites possible,because fire sites are smaller at the beginning and can be simply extinguished. The next strategy is about to choose a fire site using some priorities and then trying to extinguish buildings in that site. This is when fire sites are small enough to extinguish and are not threatening important staff in the situation such as buried civilians and agents.

The third strategy used by our team is poisoning fire in a limited area. This strategy is used when fire is too big and is going to burn the entire city and we have deduced that we cannot control it. In this situation we will define some important and critical boundaries for the fire which have to be saved and our fire agents are trying to stop the fire beyond this imaginary wall. There are also some special agents which are responsible of extinguishing fire sites that reignite after extinguishment.

Multiple properties of buildings has been used in order to decide not only about the best building to extinguish in a site but also the best site to be controlled.

8 Search for civilians

Our search task is not assigned to specific agents or specific types of agents. It is a task that is originally belong to all kind of agents and would be done automatically when agents do not have any other high priority task.

We have modeled the city's buildings into a simple graph $G(V, E)$ where V are all buildings and building i is connected to building j if it can be seen from there i.e. in the current rules it the distance between building i and building j is less than 30 meter.

Clearly , a dominating set of this graph is enough to be searched in order to be sure that all the buildings has been visited.

A dominating set for G, is $\acute{V}$ where $\acute{V} \subseteq V$ is a subset such that $\forall u \in V - \acute{V}$ there is a $v \in \acute{V}$ for which $(u, v) \in E$. And minimum dominating set is the minimum of such a subset.

As Minimum Dominating Set problems is NP-Complete[3] so there is not polynomial solution for this problem(the solution is $o(2^n)$[2]). So we need to find approximate solution. Also if map of a city is available so we can use offline process to find Minimum Dominating Set otherwise we need to use online algorithms.

- **Offline:** although we have enough time for computing MDS, It is still too much time-consuming to use exponential algorithm. There are some polynomial approximation algorithms for this purpose that can be used in an offline program to store MDS in a corresponding file for that city.One of the best approaches is **sequential greedy algorithm** which is a $n(\delta)$-constant [4][5] and polynomial time algorithm where δ denote to the maximum degree of graph. In this situation it is enough for each agent to search only the buildings in the MDS which belong to its own region in its idle time.And when it have checked all of them it can also help other agents.

- **Online:** in such a situation, agents will search their nearest unsearched building and this vertex and its neighbors will be omitted from the graph of unsearched building. Also as we discuss in offline part there are some approximation algorithms in polynomial time that can compute approximate minimum dominating set so if the number of buildings in a city is not so large we can compute MDS in the beginning of connection to the server.

9 Future works

Although Impossibles have gained an acceptable result in Osaka 2005 there are lots of weaknesses and lacks in its current code.We have to improve our message dispatching strategy which depends on solving the stated problem in the transparent communication section.

Also we have to improve our ambulance team death-time-estimation function to be more general and more independent from current misc simulator[13].This can be achieved by exerting more standard data mining[6] methods rather than manually guessing the hp-time curve.

Our FireBrigades should be completely revised in order to be more stable and simpler in comparison with other agents.

Generally we have planed to use more AI[7] and Fuzzy[8] techniques in our code to become closer to the goal of rescue simulation competitions.

[13]Currently there is a motivation among rescuers to have more fair and unpredictable simulators

References

[1] Jaffar Habibi et al:
Impossibles Team description(2005)

[2] Robson J.M:
Finding a maximum independent set in Time $O(2^{n/4})$. Technical report,I251-01,LaBRI,Universite Bordeaux(2001)

[3] TH Cormen, C Leiserson, R Rivest, C Stein:
Introduction to Algorithms Second Edition. MIT Press (2001)

[4] F. Kuhn and R. Wattenhofer:
Constant-Time Distributed Dominating Set Approximation. In Proc. of the 22 nd Annual ACM Symp. on Principles of Distributed Computing (PODC), pages 25–32,(2003).

[5] L. Jia, R. Rajaraman, and R. Suel:
An Efficient Distributed Algorithm for Constructing. Small Dominating Sets. In Proc. of the 20 th ACM Symposium on Principles of Distributed Computing (PODC), pages 33–42,(2001).

[6] Witten, I; Frank, E:
Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations Morgan Kaufmann (1999)

[7] Sutton, RS., Barto, AG.:
Reinforcement learning: an introduction Cambridge University Press (1999)

[8] GJ Klir, B Yuan:
Fuzzy sets and fuzzy logic: theory and applications Prentice Hall(1995)