FAIS, A Solution For Fire Agent Management in Rescue Simulation System

Alborz Geramifard Peyman Nayeri Reza Zamani-Nasab
Jafar Habibi

Department of Computer Engineering,
Sharif University of Technology, Tehran, Iran
(geramifa, nayyeri, zamani)@ce.sharif.edu, habibi@sharif.edu

Abstract. This paper presents a new architecture called FAIS for implementing intelligent agents cooperating in a special Multi-Agent environment, RoboCup Rescue Simulation System. This is a layered architecture which is customized for solving fire extinguishing problem. Structural decision making algorithms are combined with heuristic ones in the model, so this is a hybric architecture.

keywords: FAIS, Multi-Agent System, RoboCup Rescue Simulation, Layered Architecture

1 Introduction

The RoboCup Rescue Simulation system makes a test-bed for implementation of various Multi-Agent algorithms. Its capabilities cover a wide range of possible algorithms' styles. It is also a standard environment for testing different techniques of making standard software agents with a distributed architecture. Rescue Simulation System also provides a standard framework for testing proposed algorithms and mathematical models of disaster events.

Designing an autonomous agent set like that is required for RoboCup Simulation is quite a challenging problem. Planning effective collaboration for a Multi-Agent team in disastrous environments still remains a challenging area in AI. Efforts of Multi-Agent researches have less and more provided some standards in modeling and designing software. All these try to approach coordination between actions in different agents and making autonomous decisions that work toward the team goal. But practical results in complicated domains such as RoboCup Rescue Simulation indicates that heuristic criteria still remain as a major part of a successful system. This may signal lack of satisfactory models for these complicated situations. These evidences encouraged us towards the implementation of a hybrid system called FAIS[1]. Our structured model constitutes the core of the system which acts on advices generated by heuristic components. In fact heuristic components capture the complexity of the domain and the structured core analyzes these reasonable numbers of incoming advices and makes decision.

[1] Fire Agent Integrated System

Practical results convinced us that this is an achievement over pure structural or heuristic designs in this domain.

The next section gives a brief problem definition. Then the architecture will be introduced(Section 3) and will be examined for Fire Brigade(Section 4) and Fire Station(Section 5) agents. Section 6 concludes the paper.

2 Problem Definition

In this section we consider a simplified version of fire extinguishing problem in RoboCup Rescue Simulation. The system consists of several communicating modules: There is a standard *fire simulator* that simulates the growth and damage of fire in the simulated city, the simulated city itself is represented and managed by a software process named *GIS*. Two types of agents can help extinguish fire: *Fire Brigades* and *Fire Stations*. In our version of the problem we consider only one instance of Fire Station working. There is either a coordinating process. All other processes interact through this process. It is reasonable that we call this *Kernel*. Communication is accomplished via an extended UDP protocol named *Long-UDP*.

Fire Brigade processes can obtain visual information of their vicinity through Kernel, also they can send *move* and *extinguish* commands. A Fire Station agent can get visual information but can't send move or extinguish command. There is a limited bandwidth for communication through kernel module. Fire Station is more capable of sending and receiving messages, so -as its name indicates- can be exploited as an offline decision making agent, generating guidelines and advices for Fire Brigades. Fire Brigades also are capable of receiving damages during simulation process when they are in fiery buildings. Every time a Fire Brigade extinguishes a fire, the level of accessible water for it decreases. This simulates existence of water tank for the agent. Moreover there are certain buildings in the city that agents can repair themselves (decreasing the amount of damage) our fill water (increasing the amount of available water). These buildings are called **Refuge**s. Detailed information is available in [1].

The goal of the simulation is to build a system of agents that can extinguish the simulated city and decrease the ratio of the burnt buildings as much as possible in the simulation deadline.

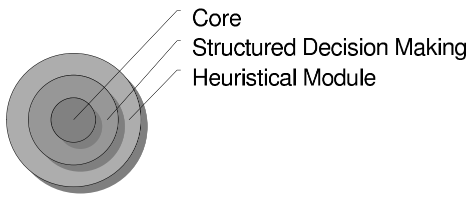

Fig. 1. Three layered architecture

3 FAIS Architecture

Figure 1 shows a schematic view of the agents' architecture. This structure is used in all agents. According to each agent task, these layers will be customized.

The system consists of three main layers. The lower level is responsible for analyzing sensing data and generating restricted number of indicator values. So the output of this layer makes the customized agent model of the world, in fact. This layer is named *Heuristic Module* in figure 1.

Decision Making layer is placed upon extracted indicators of the previous layer. The minimized world model provided by the heuristic modules is input for more structured algorithms. These algorithms then make decisions that finally will be mapped to proper actions. There are a wide range of algorithms that cover the topic of decision making. Most of these algorithms tend to make decisions upon a **restricted** number of well-defined parameters. The domain of the rescue simulation contains a large number of important parameters that should participate in decision making. Experiments for making direct decisions through passing all these parameters to such algorithms show unsatisfactory results. So, the role of heuristic components in this design is to provide an abstract view of the world model for input to decision making routines.

Finally there is a *Core* layer on top of the architecture. All decisions made by the underlying algorithms should pass through this layer. The layer acts like a filter and applies a certain database of predefined rules to input decisions. This is because structured algorithms are generally acceptable, but most times there are a set situations that those algorithms fail to behave well. The core layer is a standard place for handling these special cases. After an agent algorithm be so mature that can generally solve problems well, the database of this layer will be completed after a fine tuning step.

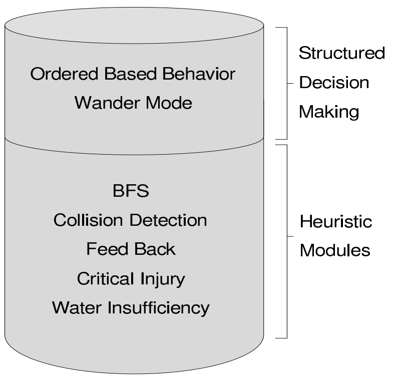

Fig. 2. Fire Brigade structure

4 Fire Brigade Design

We are going to introduce the details of the previously three-layered architecture in the Fire Brigade agent.

Fire Brigade lower layer consists of the following modules:

BFS: This is the system's path planning module. Typically the agent needs to find physical paths on the simulated city to change its position. The input to the problem is a graph. Graph edges represent the open roads at the beginning of the disaster simulation. As the time passes, some roads will be opened so this graph is a dynamic graph. The problem of handling such a dynamic graph is solved partially by deterministic approaches like continuous Dijkstra[2]. The base of our solution here is a simple BFS over a graph of all the potentially open roads. As the agent explores the graph, closed roads are discovered and marked on the graph.

Collision Detection: Sometimes in the graph there are open roads but agent can't travel all the paths. This occurs for example when a certain direction of the road is not accessible because some other agents have blocked it. In this situation this heuristic strategy is used: Upon discovery of a blocked direction of a road, the certain edge will be removed from the graph, temporarily. After a fixed period of time, the edge returns back again.

Feedback: At certain time intervals, the agent reports some important information that discovers when exploring the city, to the Fire Station agent. These complementary information are the base of what is known as station's world model.

Critical Injury: When an agent receives a certain minimum level of damage, it ignores all plans except going to Refuge buildings.

Water Insufficiency: When Fire Brigade detects that it has ran out of water, all plans will be ignored else going to the nearest Refuge building.

Figure 2 shows the total view of these components in respect to the total architecture. The following components are deterministic decision making modules in the Fire Brigade structure.

Ordered Based Behavior: This is a component that makes the main agent decisions. This sub-layer works according to an advice based structure[3]. The Fire Station processes world model which is mostly provided by the feedbacks sent by the Fire Brigade, and generates the advices that Fire Brigades obey in order to remain coordinated.

Wander Mode: After Fire Brigades extinguish the fire, they start to explore the city regularly. This shows as an effective method, because extra information about the city will be discovered through the remained time till the end of simulation.

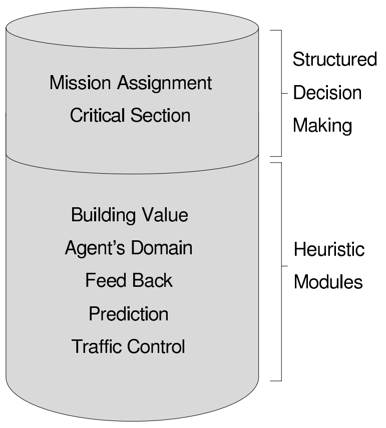

Fig. 3. Fire Station Design

5 Fire Station Design

Fire Station module is responsible for centered decision making and coordination. It computes a profitable subset of fiery buildings and assigns a subset of idle agents to them for extinguish operation. The system sets a proper timeout for assigned mission, so after timeout, Fire Station considers the assigned Fire Brigades as free agents. Fire Brigades also will abandon the mission if the timeout expires. Figure 3 shows the details of the Fire Station. The lower layer components are:

Building Value: This is a heuristic real value computed for each building. Buildings with large amounts of this value are candidates for extinguishing in the mission assignment. The value for building b is computed in the following way:

$$V(b) = \beta N(b) + \gamma HV(b) - \alpha \sum_{i \in J} d_i(b) \tag{1}$$

$d_i(x)$ is the distance between Fire Brigade agent number x and building b. J is the set of indices of currently free agents. $N(x)$ is the number of unfired neighbors of the building x. $HV(x)$ is a value associated with a building x that indicates how much x can be destructive in respect to the other buildings. For example, a fiery building in the margin of the city is much less dangerous than a fiery building in the city center. Figure 4 shows the details. First all fiery neighbors are computed, and the convex hull of the set, H, is constructed through a Graham scan method[4]. Then a unit vector $\mathbf{u}$ is computed such that in position of building x, it points toward bisector of angle of H at vertex x (see figure 4). All *other* fiery buildings that are not much farther than a certain margin, from the semi-line in direction of $\mathbf{u}$ starting at x, are examined to find the nearest one to x, let y. We define

$HV(x) = distance(x, y)$. α , β and γ are positive factors that are determined after fine tuning of implemented agents.

So the intuition behind (1) is that buildings with more unfired neighbors, having more dangerous position to spread fire and with much free Fire Brigades near them are more profitable to choose.

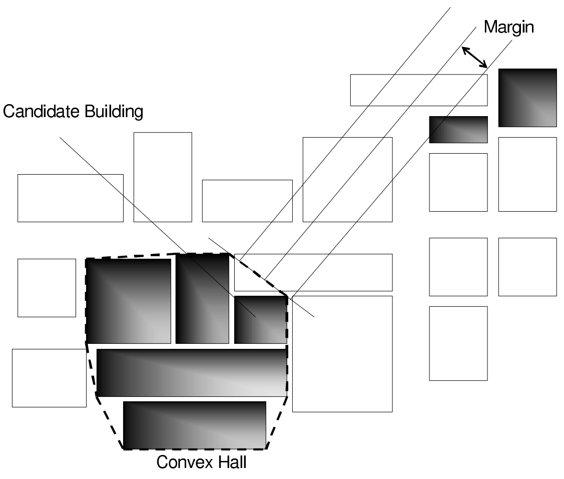

Fig. 4. Finding the fire border

Agent's Domain: For the previous section, we should compute the set of free agents for each building. But this differs in respect to each building because the roads graph is dynamic and some idle agents can't reach some certain buildings (that is some certain graph vertices). So, in this component, the set of reachable buildings for each agent is computed through a BFS circumnavigation. Therefore, the set of free agents considered for each building can be a *proper subset* of all idle agents.

Prediction: The Fire Station needs to have proper approximation of future of the system to schedule effective missions. As phenomena like fire spreading in deterministic (previously known) environments are less and more obey some certain rules, it is possible to provide some approximation. This module is responsible for maintaining a compact set of previously fire spreading scenarios and matching the current scenario with them to predict some future parameters like which buildings will be ignited in the next cycles and how much water in sufficient for extinguishing a certain building. A feed forward neural network provides the compact representation in addition to proper response time and accuracy [5].

Feedback The process of fire spreading is dependent on the current status of the city. For example broken buildings are more vulnerable to fire and so spread the fire more quickly. The predictor must be informed of such information in order to predict properly. In fact **Feedback** module of Fire Brigade provides discovered information as agents explore the city, and in this peer module, Fire Station predictor applies the received information in prediction module.

Traffic Control: When agents are assigned missions, in many cases the roads are blocked because of working agents. As agents are working independent, it is very difficult to resolve the problem only by means of Fire Brigade agents. This module of Fire Station monitors the assigned missions and makes proper advices for each agent in order to avoid blocking the roads, as much as possible. In many cases, sending agents to buildings near the target building is more appropriate than keeping them in the roads. Figure 5 shows a sample of this event before using Traffic Control module (left) and after applying Traffic Control rules (right).

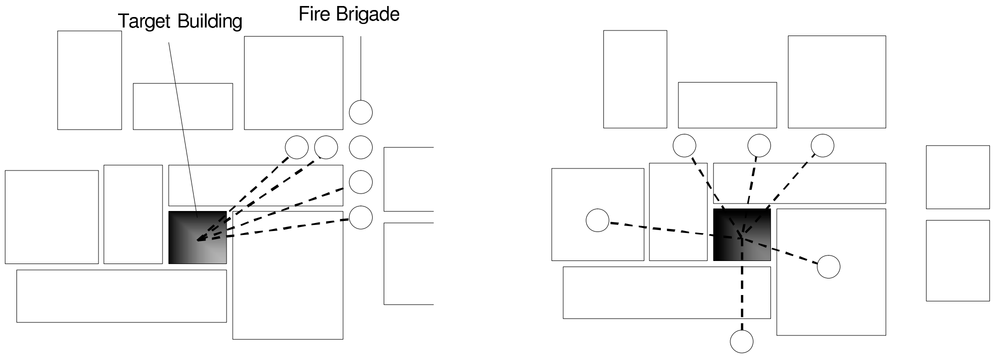

Fig. 5. Fire Station Traffic Control

The structured decision making components are:

Mission Assignment: This module does the real task of mission assignment. First prediction module approximates future state of the scene and then potential mission buildings are identified. Building Value module then selects some profitable buildings between these. This comes up with a limited number of proper buildings. Free agents should be assigned to these buildings in an optimum manner. Here, from the optimum manner we mean that the sum of distances for all agents to reach their mission locations be minimum. The problem is a restricted type of the graph matching problem with limited degrees at vertices [7]. An LP solver does this final assignment [6].

Critical Section: Extinguishing fire in the first cycles of the simulation is of special importance. The more buildings be extinguished at first, the less fire spreading occurs and handling the scene becomes easier. In addition in the first cycles, prediction module has not enough information to make satisfactory predictions. Even if the prediction results are acceptable, many roads are blocked at the start of simulation,and so many assigned missions will not be completed successfully. Critical Section component overrides mission assignment in the certain interval of initial cycles. In this interval, missions are not assigned optimally, but simply all agent are assigned to the best building for extinguish. This has showed a better performance than using prediction and assignment at the beginning.

6 Conclusion and Results

A set of fire agents (Fire Brigade, Fire Stations) based on FAIS architecture is fully implemented and participated as Arian team in the 2003 Rescue Simulation Robocup Competitions. There was a wide range of strategies available in the competition and FAIS showed the best performance between them. Table 1 is the result of the round one of the competition.

ARIAN	62.35
S.O.S.	56.08
The Black Sheep	43.83
YowAI2003	41.30
Eternity	38.74
POLITECS2003	31.70
NITRescue03	29.56
RESQ FREIBURG	28.58
SBCE_SAVOUR	28.18
RAYAN	27.74
RoboAkut	25.28
PAKRescueTeam	23.50
SBCE_RES	22.71
Ferdowsi	14.88
UVA RESCUE C2003	13.56
ToosRes	13.50
BanzAI	10.12

Table 1. RoboCup 2003, Rescue Simulation Competition results, Round one

References

1. The RoboCupRescue Technical committee, RobocupRescue Simulator Manual
2. J. S. B. Mitchell. Shortest paths among obstacles in the plane. Internat. J. Comput. Geom. Appl., 6:309-332, 1996.
3. M. Ahmadi, M. Motamed, J. Habibi , Arian: A General Architecture for Advisable Agents, International Conference of Machine Learning(MLMTA'03), Las Vegas, Nevada

4. F. P. Preparata and M. I. Shamos, Computational Geometry: An Introduction, Springer-Verlag, 1985
5. Laurene Fausett, Fundamentals of Neural Networks, Prentice Hall International, 1994
6. B.H. Korte and Jens Vygen, Computational Geometry: An Introduction, Springer Verlag
7. C. Berge , Graphs and Hypergraphs, 2nd rev. ed. Amsterdam, Netherlands: North-Holland, 1976