Arian Agents: A Set of Implemented Agents for RoboCupRescue Simulation Environment

Jafar Habibi Mazda Ahmadi Ali Nouri Mayssam M. Nevisi
Alborz Geramifard Peyman Nayyeri Mayssam Sayyadian
Hassan Khaleghi Mehran Motamed Reza Zamaninasab

Department of Computer Engineering,
Sharif University of Technology, Tehran, Iran
arian@ce.sharif.edu http://ce.sharif.edu/robocup/arian

Abstract. Arian agents are a set of implemented agents for RoboCupRescue simulation environment. They ranked first at RoboCup 2002 competitions at Fukuoka. The main reason for Arian's success is its flexible architecture. In this paper, we introduce Arian agents' overall architecture.

1 introduction

Recently, multi-agent systems have gained a lot of attention. Robotic soccer [2], intelligent control of multiple agents, automated driving (e.g. [1]) , and e-commerce agents are examples of challenging multi-agent domains.

RoboCupRescue Simulation with heterogeneous agents with different abilities and of course responsibilities with a limited communication, is an excellent framework for multi-agent planning, communication techniques, coalition formation and task allocation.

Arian was the champion of the RoboCup2002 rescue simulation league in Fukuoka and won the second place in the RoboCup2001 in Seattle. They have done different works in agent communication languages[5], coalition formation[3] and agent architectures[4].

2 Arian Architecture

The overall architecture of Arian agents is presented in figure 1. Each module is independent of the others and can be implemented separately. As we will see, the main benefit of using Arian as the basis of developing agents for RoboCupRescue simulation environment, is that the researchers can only design the decision making module. In the rest of this section we will briefly describe Arian architecture's modules.

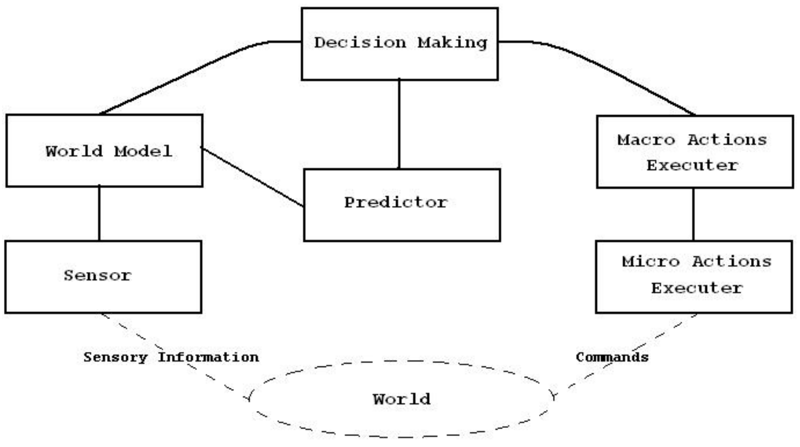

Fig. 1. Arian agents' overall architecture.

2.1 Sensor

As mentioned before the communication protocol used in RoboCupRescue system is based on UDP. The sensor module facilitates communication between server and agents with the help of a simple interface. This module parses the received sensory information.

2.2 World Model

This module provide an interface for manipulation and extraction of high-level information from a world model database. This world model database is updated in the beginning of every cycle. The class hierarchy of objects in the Arian's world model is shown in figure 2. Each of these objects has the properties of the corresponding objects in the world.

2.3 Predictor

Given a macro action, a world state, and a prediction time, this module predicts the state of the world in the given time. This module works based on hundreds of previous runs and statistical methods.

2.4 Macro Actions Executer

A macro action is an action that lasts more than one single cycle. The macro actions in current version of Arian is as follows:

– **Extinguish a Building till totally extinguished:** In this macro action, the agent spreads water on the specified building with maximum power until the building gets extinguished.

- **Clear a long Road:** In the simulated world, each long road consists of a number of short roads. In each cycle an agent is able to clear only a short road, so clearing a long road takes more than one cycle.
- **Rescue a Civilian:** Rescuing of a civilian consists of several primitive actions that this module takes care of. First, the civilian should be taken out of the building he is in, then it should be transferred to an ambulance and the ambulance should take the civilian to the closest refuge and finally, drop him there.
- **Explore the City:** In many situations, agents tend to explore the city in order to get the most possible information about the world. This module handles this matter with some graph based algorithms.
- **GotoXY:** By executing this macro action, the agent tends to move to another position that can probably take more than one cycle. There can be different algorithms for finding a path between two positions. In Arian agents there are two implemented algorithms for path planning, one based on the A* algorithm and the other on the Floyd algorithm. These methods get current and final positions and return a route between these two positions. Although these algorithms may not be optimal, but are useful for those whose research focus is not path planning.

2.5 Micro Actions Executer

This module handles communication with the kernel and sends commands to the kernel. For each of the legal commands that agents can send to the kernel, there exists an interface in this module.

2.6 Decision Making

Usually this module is the one that most researchers are interested in designing and implementing it. Researchers can use our architecture and just design a decision algorithm.

So far we faced the decision-making problem as a riddle, which can be best, completed by an optimum heuristic function, which is based on CBR and Decision Tree but from another aspect we can see each agent decision making algorithm as a gene so we can also apply evolutionary methods to optimize their behavior. Respectively we also have a total value function which computes the value of the hole process of the agents in total cycles so we can set this as our base function for optimization.

Implementation of the decision algorithms can be one of Finite State Automaton, Binary Decision Diagram or Neural Networks; but in this specific environment that we probably tend to enforce some heuristic functions, we better use BDD as the implementation of the genes because our state space is so huge that neural network implementation will fail to converge in a logical period and from another aspect experiences show that in such situations BDDs do far better than DFAs and converge faster.

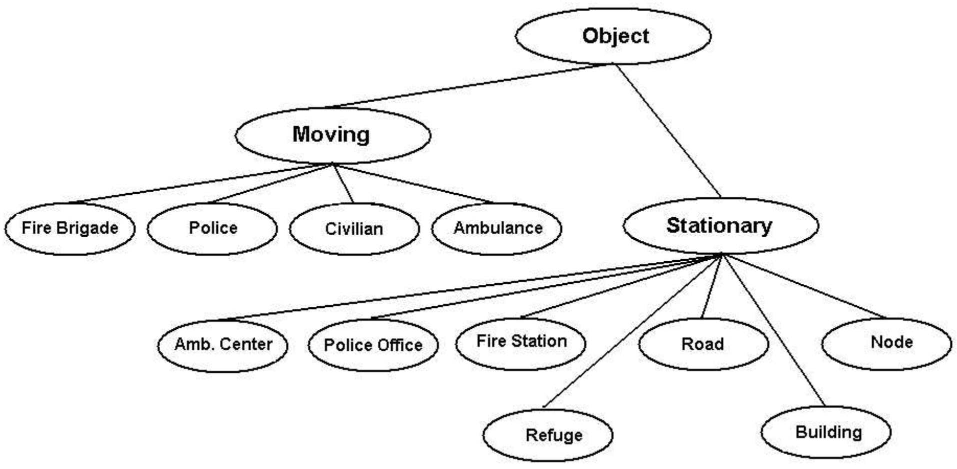

Fig. 2. Objects' hierarchy in the world model. The nodes represent objects and two nodes are connected if the lower node inherits the upper one.

References

1. A. Deshpande, D. N. Godbole, A. G. and P. Varaiya, "Design and Evaluation Tools for Automated Highway Systems", Hybrid Systems pp. 138-148, 1995.
2. I. Noda and P. Stone, "The RoboCup Soccer Server and CMUnited Clients: Implemented Infrastructure for MAS Research" Journal of Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publisher, 2002.
3. M. Ahmadi, M. Sayyadian, H. R. Rabiee, "A Coalition Formation for Task Allocation via Genetic Algorithms", In Proceedings of The First Eurasian Conference on Advances in Information and Communication Technology (EurAsia ICT 2002), LNCS, Springer, 2002.
4. M. Ahmadi, M. M. Nevisi, "An Architecture for Multi-layred learning in Autonomous agents", Proceeding of FSCCSI, Isfahan, Iran, 2002 (in persian).
5. M. Ahmadi, M. Sayyadian, J. Habibi, "A Learning Method for Evaluating Messages in Multi Agent Systems", In proceedings of the Agent Communicatin Languages and Conversation Policies, AAMAS'02 Workshop, Italy, Bologna, 2002.