PAKRescueTeam's Description

Sébastien Paquet and Brahim Chaib-draa

DAMAS laboratory, Laval University, Canada
{spaquet;chaib}@damas.ift.ulaval.ca

Abstract. A fundamental difficulty in cooperative multiagent systems is to find how to efficiently coordinate agents' actions in order to enable them to interact and achieve their tasks proficiently. One solution for this problem is to give the agents the ability to learn how to coordinate their actions. In this paper, the PAKRescueTeam's agents are describe and three learning methods used by those agents are presented.

1 Introduction

In multiagent environments with heterogeneous agents, cooperation between such agents becomes a really important issue. In this type of environment, there is no agent that has all the required capabilities to accomplish its goals by itself. Each agent needs the cooperation of other agents to be efficient.

This paper presents some learning methods used in the PAKRescueTeam team to improve the agents' coordination, but before that, it begins by presenting the important characteristics of the RoboCupRescue environment which motivate our approach.

2 RoboCupRescue: a Complex Environment

Some of the difficulties or the major issues in the RoboCupRescue environment are: agents' heterogeneity, long-term planning, emergent collaboration and information access [1]. The first difficulty comes from the fact that the agents are heterogeneous, they cannot do everything by themselves, therefore, they need to cooperate in order to accomplish their goal efficiently.

They also have to make long-term plans, in which they decide on the most urgent problem and what their next course of actions will be. This is very difficult since agents cannot anticipate what the situation will be like in the future.

The emergent collaboration is an other major issue, because agents will have to collaborate but this collaboration has to be emergent or evolutionary to give enough flexibility. The situations in which the agents have to work are very different so it would be interesting if agents could learn to adapt their collaboration strategies to the current situation.

One of the major problems in disasters is the difficulty in getting accurate information within reasonable time. Agents have very bad access to information, they only have local perceptions, so they need to communicate accurately to have a better assessment of the global situation and make the correct choices. The coordination of all agents depends a lot on the accuracy of the communication between those agents.

3 Learning Coordination

As previously stated, the agents in RoboCupRescue have limited but complementary capabilities, so there are situations where they need to coordinate their actions in order to be efficient. There are many solutions for the problem of coordination and they can be divided in three general classes [2]: those based on communication, those based on convention and those based on learning.

Learning is an interesting approach because it removes from the designer the hard job of defining all coordination procedures required for all possible situations. Instead agents learn how to improve their coordination by interacting with the other agents. The learning approach is really promising for the RoboCupRescue simulation since it can ensure a good coordination at all times, even though there are a lot of possible scenarios. In addition learning enables the multiagent system to adapt itself gradually if the environment changes. This section presents three learning approaches to improve agents' coordination in the RoboCupRescue environment.

3.1 Learning How to Communicate Efficiently

One of the major issues in the RoboCupRescue simulation is the communication, it is limited, uncertain, but necessary. Agents only have local perceptions, so they need to communicate to have a better situation assessment and to obtain and maintain an efficient coordination. On the other end, communication has a cost and in the simulation there are also some limitations on the number of messages an agent can send and receive.

The coordination of the agents' actions is really dependant on the communication. Agents have to communicate to know what the others are doing, to notify others what to do, etc. Consequently, if we want to improve the coordination between agents, it is a good start to try to improve the efficiency of the communications.

To achieve that, we could enable agents to learn, over some simulations, which messages are really useful and which ones are not. With this information, agents will be able to send and read only the more important messages. To learn that, agents can analyze their execution trace [3, 4], at the end of the simulation, to learn which messages are the most important to send and listen to. For each time step, this trace contains for each agent some information describing the situation in which the agent was at that time. At the end of the simulation, the agents will meet to analyze their traces to see which messages were really useful and which could have been ignored. It is like real fire fighters analyzing after a training exercise, what as been done well and what could be improved.

To make this analysis, each agent will send all the messages it could have sent to all other agents and ask them to return a utility evaluation for each message. The utility value returned by an agent depends on: the information contained in the message, the position, the activity of the agent, etc. For a fireman, for example, a help request to extinguish a fire is important if the agent is close enough to go and extinguish it in time.

Suppose agent A has received from other agents all the utility values for the message m_1, it makes an average of those values. This average gives an indication of how much the message would have been useful for all other agents. With this information, agent A updates its knowledge base to adjust the utility value of the message m_1.

What agents can gain from this meeting? If we look at the sender agents, they will have an indication about which messages are useful for the other agents. Therefore, they will be able to evaluate if a message is important enough to be sent. In addition, receiver agents will have a better idea about which messages to listen to and which ones to ignore. If an agent receives only useless messages from another agent, it will reduce the probability of listening to this agent.

3.2 Learning how to Behave in the City

Another learning opportunity is to learn how to react in a specific city after a disaster, like a big earthquake. Since agents are acting in the same city over and over again, they could learn the best way to manage a disaster depending on which sectors of the city are having problems. For example, they can try to start fires in one part of the city and see what is the most appropriate approach to manage the problem they have created. For instance, they can see which roads are the most important to clear, how to position fire fighters or how to organize teams of rescue agents, all of this, depending on where the disaster is occurring in the city.

To learn which are the most useful roads, many different situations can be tested. For example, many fires can be started in one defined sector on the map, with all roads cleared, and agents will do their best to extinguish those fires. During their execution, they will record which roads they are using. At the end of a simulation, we have for each road the number of times it has been used. This information could then be used to classify the roads by their statistical importance. Consequently, *PoliceForce* agents will have an ordered list of roads on which they can rely to choose which roads to clear if a fire starts in the same part of the city.

3.3 Anticipation

The last learning opportunity presented in this paper is to enable agents to anticipate their actions and other agents' actions. For instance, if a *FireBrigade* agent is currently extinguishing a fire and it estimates that the fire will be extinguished in ten minutes, then, it could already estimate its future destination and ask a *PoliceForce* agent to clear the roads between the current fire and the next one to attend. Of course, if we want agents to anticipate effectively we have to learn how the disaster evolves in time and how the agents' actions interact with the environment. This could be done by observing the evolution of a certain number of simulations in one given city, to be able to anticipate such evolution in future simulations.

For instance, if we want *FireBrigade* agents to learn how much time it takes to extinguish a fire, we could test on some simulations in how much time fires were extinguished. To do that, we can test the time needed on different situations, i.e. with different values for characteristics such as: the number of *FireBrigade* agents, the building's size, the building's fieriness, etc. This method enables agents to estimate the time needed to extinguish a fire. With those estimation they will be able to construct more accurate long-term plans. Those plans could then be used by agents to anticipate the actions of others and make actions that will help the agent in the future.

Another way to anticipate would be to enable agents to learn a model of the other agents. Since there are only three moving agent types, each agent will only have to learn two models. The model cannot be unflexible because each agent is learning, so the model will evolve after each simulation. At the end of each simulation, each type of agents can give to the other agents' types their utility function. For example, a *FireBrigade* agent can give to the other agents the function it uses to choose which fire to extinguish. With this information, *PoliceForce* agents will be able to anticipate, without any messages, which building the *FireBrigade* agent will extinguish next and to help it get there, it could clear the roads in advance.

4 Conclusion

This paper has presented three learning approaches to improve agents' coordination used by the PAKRescueTeam team in the RoboCupRescue simulation. One of the goal of the RoboCupRescue is to be a good test-bed to test multiagent approaches and it is in that optic that the approaches and the algorithms have been developed for our rescue team. It is an ongoing research project at our laboratory to design and test some learning algorithms that will be well suited and useful for complex multiagent real-time systems like the RoboCupRescue.

References

1. Kitano, H.: Robocup rescue: A grand challenge for multi-agent systems. In: Proceedings of ICMAS 2000, Boston, MA (2000)
2. Boutilier, C.: Planning, learning and coordination in multiagent decision processes. In: Proceedings of TARK-96: Theoretical Aspects of Rationality and Knowledge, De Zeeuwse Stromen, Hollande (1996)
3. Sugawara, T., Lesser, V.R.: Learning to improve coordinated actions in cooperative distributed problem-solving environments. Machine Learning **33** (1998) 129–153
4. Garland, A., Alterman, R.: Learning to better coordinate with autonomous agents. Technical Report CS-01-219, Brandeis University (2001)