RoboAkut Rescue Team

Bilge Talaysüm and H. Levent Akın

Department of Computer Engineering, Bogazici University TR-80815 Bebek, Istanbul TURKEY

talaysum@boun.edu.tr, akin@boun.edu.tr

Abstract. RoboAkut rescue team combines reinforcement learning with cooperation among agents so that agents can learn mutually from each other. There is a hierarchy among the agents so that agent coordination is centralized. Each agent is capable of deciding for itself when in lack of support from other agents. The agents make use of the sensory information they obtain to learn the state of the environment, and use this information to decide on the actions to perform. Each agent learns from the results of the actions it does. The platoon agents are capable of sharing information with each other when necessary. With limited communication possibilities, the agents try to use as small number of messages as possible. The agents can do both greedy actions, and also some exploration rate can be introduced so that the agent tries other actions than the one it decides to be the optimal one. Exploration is used during training and greedy actions are used during competition.

1 Introduction

The members of RoboAkut are Assoc. Prof. H. Levent Akın, and Bilge Talaysüm, a graduate student from the department of Computer Engineering in Bogazici University. One of the main research areas of the Artificial Intelligence Lab of the department is multiagent design. Several projects are carried out in this area including design of robots for RoboCup League. Coinciding both with the recent disasters we have lived and our interest in multiagent design we have decided to participate in the RoboCup Rescue Simulation League. The design and implementation of a rescue team consisting of autonomous and cooperating agents is carried out as a master's thesis subject. RoboAkut has both central and platoon agents of firefighter, police officer and ambulance.

2 The General Structure

Both the center agents and the platoon agents have the following components: World model, learning component, I/O component and performance component. In addition, the platoon agents have a route finder component. These components are described in the following sections.

2.1 The World Model

Each agent keeps a world model of its own. Initially, the simulation kernel provides all the structural information about the world. With this information, an agent can move to any destination in the environment if none of the roads are blocked. In cycles later than cycle two, the kernel only provides visual information within 10 meters of each agent. Thus, the agent cannot keep up to date world information by storing only the visual data provided by the kernel. An agent can also hear the other agents within 30 meters. By intercepting such messages , an agent can improve its knowledge range.

Other than keeping raw world information, which helps in locating targets, calculating distances and routes, each agent also keeps an abstraction of the world in order to describe the environment's state by a single index. Each time the world information of an agent is updated, the agent counts the number of burning buildings, burnt down buildings, injured civilians, dead civilians, and blocked roads. With these numbers, the agent calculates three indices, which separately describe the severity of the conditions of the buildings, the civilians, and the roads in the environment. The indices are in the range [0, 1], where zero describes an ideal state where everything is under control, one describes a disastrous state. These indices are used by the learning component to calculate an overall index for the state the agent believes to be in.

2.2 The Learning Component

Both the platoon agents and the dispatcher agents use reinforcement learning. The job of the learning component is to calculate the abstract state of the environment as described in section 2.1, to learn from feedback by assigning values to state-action pairs (Q-values), and to decide on an action by considering Q-values. The Q-values are updated using temporal difference method [1], [2].

For a platoon agent, what is referred to as an "action" is to "go to target X". The identification of the target is done by choosing between different alternatives for action. The action alternatives for a platoon agent are: Go to the closest target, go to the farthest target, go to the target which is the most severe, go to the target which is lightly damaged, and go to any target. The platoon agent updates its Q-values when it has decided its own action, without a command from the center, and it has either saved its target or if the target was ruined. The reward value associated with a state, and is used to update the Q-values, is more positive if the overall state index is near zero, more negative if the overall state index is near the maximum value. Thus, the agent gets positive reward if it is able to keep the world in an ordered state, negative reward if the environment is breaking down. Naturally, in a multiagent environment the state is determined not only by the actions of one agent, but also the success or failure of other agents. Therefore, the start and end states of an action are not deterministic. This nature of the environment makes temporal difference of Q-values the most suitable reinforcement learning technique to be used for this task.

An "action" for a dispatcher agent means assigning agents to targets. There are two assignment methods for a dispatcher agent: Assign each agent to the target closest to it or assign each job to the closest agent, starting by the most urgent target. A dispatcher agent updates its Q-values by getting feedback from its platoon agents. If most of the agents were successful because of the assignment, the Q-value of corresponding state-action pair increases. Different types of actions can be added for both central agents and platoon agents without disturbing the learning mechanism.

The agents can be run and decide on their actions in two different modes: greedy or explorer. If the agent is running on greedy mode, it always tries the action with the best Q-value so far. If the agent is running in explorer mode, it can accept actions with lower state-action values with a predefined probability, which is the exploration rate. The explorer mode is to be used during training, and greedy mode is to be used during real action.

2.3 I/O and Cooperation

The I/O module is separate from other modules to be able to isolate the effects of the possible changes in the structure of the interactions with the environment from other modules of an agent. The I/O module contains the raw functions for sending and receiving messages to the kernel, and wrapper functions used for sending special messages.

Communication is done when the dispatcher agents assign jobs, the platoons request jobs or send results of jobs. If an agent is assigned a job but it cannot get there because the roads are blocked, it must report this case to the police officers, which in turn either drop their other jobs to clear the indicated road, or put the request into queue. It was seen during sample runs that the major difficulty in the environment is the blocked roads. Therefore, the communication described has an important effect on the success of agents. In addition, if an agent has not filled its quota of communication for a turn, but finished executing what it can do, it can also tell sensory information to others in order to be able to keep everyone's world information up to date.

2.4 The Route Finder Module

The platoon agents are equipped with a route finder module. The information provided by kernel during the first two cycles of the simulation provides the information necessary for finding paths to any target in the environment. The route search is done as a breadth first search and loops are prevented by not adding a node that has been expanded before to the list of nodes to be expanded. Since the agent cannot always move to the target in one turn, at each loop, if the agent has not arrived, the route is re-calculated.

2.5 The Performance Module

The performance module of the agents executes in a loop and controls the flow of actions. The loop starts with receiving messages from the kernel, and making actions upon the received information. Two types of messages can be received by each agent "sense" message or "hear" message. Each "sense" message sent by the kernel to the agents indicates the start of a new cycle. The agents keep track of the elapsed time and the available number of messages that can be heard or said in a cycle by keeping track of the received "sense" messages.

The platoon agents update the world information, abstract world information and the internal state representation in each cycle. Targets indicated by other agents have priority over targets indicated by center. The agents own decisions have the least priority. If a job has finished and that job was commanded to the agent by the center, the agent sends feedback to the center indicating the success or failure of the job, upon which the center updates the corresponding Q-value. If the action was the agents own decision, the agent's Q-value is updated. When agents are stuck, they ask roads to be cleared, except for police officer agents. They go to their target by clearing blocked roads on the way.

In each cycle a dispatcher agent makes assignments of jobs to agents. The agent keeps information on which agent is assigned to which job and which assignment method was used for those assignments. This information is used to update the state-action values of the center agents.

3 Conclusions

Our rescue team was developed considering the changing conditions, the necessity to act fast and limited communication means in a disaster environment. We have built a model that incorporates mutual learning and cooperation, in order to develop more flexible agents that behave better in a disaster situation.

Acknowledgements

This project is sponsored in part by Bogazici University Center for Disaster Management (CENDIM).

References

1. Norvig, P., Russel, S.: Artificial Intelligence: A Modern Approach, Prentice-Hall, 1995.
2. Sutton, S. R., Barto, A. G.: Reinforcement Learning I: Introduction, MIT Press, Cambridge, MA, 1998.