S.O.S. Team Description

Shahab Javanmardy, Arash Rahimi, Hamideh Vosoughpor,
Vahid Ghafarpour, Haamed Gheibi, Mohsen Izadi

Javan Robotics Club,
Javan Farhangsara, Khavaran Ave., Tehran, Iran
{shahab, arahimi, hvosough, vghafarpour,
hgheibi, mizadi}@Javanroboclub.com
http://www.javanroboclub.com/SOS/index.html

Abstract. In RobocupRescue Simulation, Huge environment, wide variety of parameters affecting the decision process and limitations applied to inter-agent communication are important problems that a team's agents should overcome to perform well during the simulation process. In this paper, we described some of our solutions to these problems which are our special agent design, knowledge reasoning methods, low-level skills and cooperation methods

1 Introduction

In Robocup Rescue Simulation environment, achieving the main goal (which is preventing fire from spreading over the city and saving as many civilians' lives as possible.) requires high-performance cooperation of several heterogeneous Agents.

We believe that a high-performace cooperation, first requires each agent to have well-designed low-level skills and then a good strategy based on these low-level skills for cooperating together. So during the Agent design process we focussed on designing :

1. A suitable and flexible agent architecture capable of defining high-level strategies easily.
2. Efficient methods for reasoning about world model data in the best way possible and designing powerfull and fast low-level skills and complex actions.
3. Cooperation methods and algorithms optimized for Rescue Simulation environment.

In this paper, we'll have a short overview on each of the subjects above.

2 Agent Architecture

The architecture we've designed and implemented for our agents is a "State-based" architecture, which means that several states have been defined for each

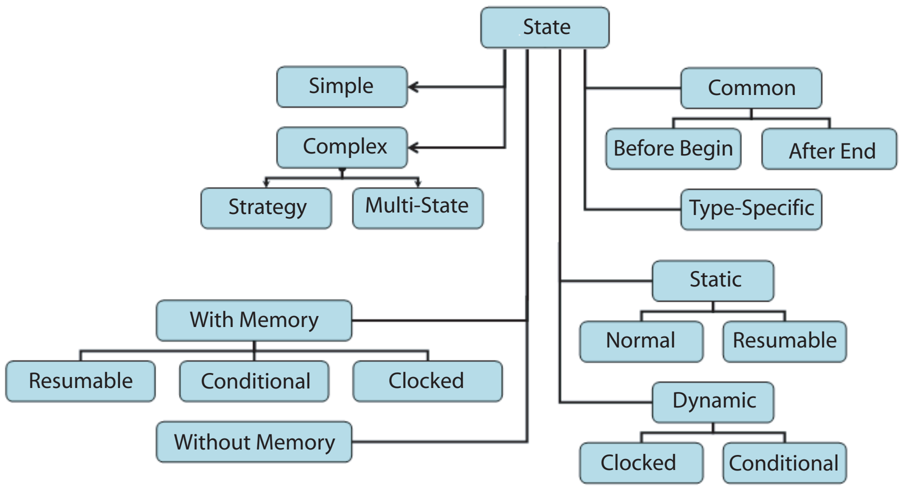

Fig. 1. State Classification Diagram

agent of a specific type, due to possible conditions of that agent and the tasks that it should perform during the simulation.

An agent performs a specified task while being in a state and also listens for "Events" which may occur during the simulation. An event causes the agent to perform a certain job and sometimes changes the state of the agent.

We've classified "Agent States" due to several criteria into certain groups. For example in a certain classification, we've divided states into Normal, Conditional and Clocked States. A normal state is only changed by an external event but a conditional state is changed if a certain condition is satisfied and at last a clocked state is a state that automatically changes itself after a certain period of time.

There are also many other classifications shown in figure 1.

In addition to the state-based agent architecture for defining deterministic behaviour for our agents we also added an additional decision thread in which we apply high-level algorithms, statistical analysis, online learning algorithms and generating unexpected events for State Machine layer.

Layers of the architecture and relations between these layers are shown in figure 2.

3 Pathfinding

Pathfinding is an example of agent low-level skills which is of remarkable importance as it is inevitable for agents to move to several places during the simulation and it is important for the agent to move to these places along an efficient and minimum-cost path.

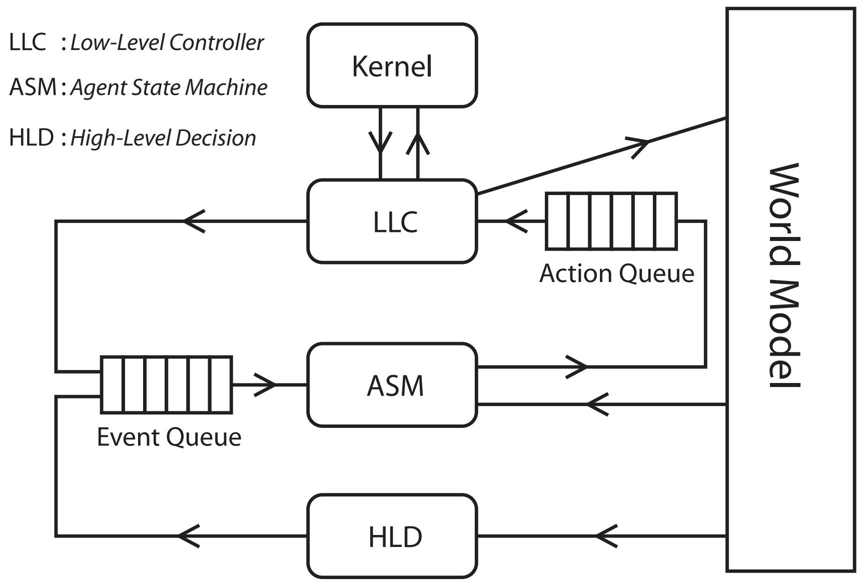
LLC : Low-Level Controller
ASM : Agent State Machine
HLD : High-Level Decision
Kernel
LLC
Action Queue
ASM
Event Queue
HLD
World Model

Fig. 2. Agent Architecture Diagram

For pathfinding we use two seperate methods. One method we are currently using is based on “Focussed D* Algorithm”[1]. D*(Dynamic A*) algorithm plans optimal paths in real-time by incermentally repairing paths to the agent state as new information is discovered. Focussed D* is a special extension of D* that focuses the repairs to significantly reduce the total time required for the initial path calculation and subsequent replanning operations.

The other method we are currently working on, and is not completed yet is based on a method called “Radar Path Planning”[2]. In this method a path is built in the following way : In regular intervals the target point sends out a wave front traversing the work space with constant speed. The neighbor point of the starting point that is first reached by the wave front must be part of an optimal path. This neighbor point becomes now the new starting point and the procedure is repated until the target point is reached.

When the source point, the target point or an obstacle moves, a new optimal path is dynamically build. The path is cut and rebuild when an obstacle moves into the path. When a path becomes sub-optimal by a moving target point or obstacle, a new optimal path is found by the path length limitation and cutting noise.

Here, we’re working on an implementation of this method using “Spiking Neural Networks”.We’ll have a seperate paper about this method if we can finish it and gain successfull results.

4 Cooperation between Agents

Once we've completed designing agents with powerful and flexible behavior, the main problem is an efficient way of cooperation between agents. Here as the number of messages each agent can say or hear in a step is limited to a small number, this problem becomes more complex. Due to this problem, we've divided the task of optimizing inter-agent cooperation into two parts:

1. Optimizing communication method
2. Designing cooperation algoritms which are less dependent or independent of communication.

The first solution, itself is divided into two parts. One part is designing a high performance communicatoin protocol. So in designing process we focused highly on "Message Content Efficiency" which means that higher rate of $\frac{meaning}{length}$ of messages was important to us. We also made the communication protocol in such a way that in many certain situations, our agents can predict contents of a message just by knowing the ID of the sender.

The other part of this solution is "Agent Grouping" and making use of indirect communication. In agent Grouping our agents are divided into several groups, each of which is consisted of at least four agents. Each group has one or two "Group Leaders" which are responsible of ex-group communication. These agents gather and manage data received from other group members and transmit useful data to ex-group agents. Grouping decreases the number of messages transmitted to important agents and dramatically increases the $\frac{meaning}{number\ of\ messages}$ rate.

In the second solution our focus was on designing and implementing methods for automating the process of inter-agent cooperation using predefined scenarios and strategies. For instance one idea is making parties of heterogeneous agents and defining each agent in a party, a set of tasks to do, in order to achieve a predefined goal assigned to the party.

For example one considerable party is a party of two fire brigades and a police force in which fire brigades decide on a building to extinguish and say it to police force. Then the police force finds and clears a route to that building while fire brigades follow it.

References

1. Antony Stentz. *The Focussed D* Algorithm for Real-Time Replanning.* In International Joint Conference of Artificial Intelligence, 1995.
2. Ulrich Roth, Marc Walker, Arne Hilmann, and Heinrich Klar. *Dynamic Path Planning with Spiking neural Networks.* In International Work-conference on Artificial and Natural Neural Networks, 1997.