CS886: Multi-Agent Systems

Project Three: Robocup Rescue

YabAI - Winner of the First RoboCup Rescue Competition

Michael Jarrett (99318764)

July 7, 2004

Abstract

RoboCup Rescue Simulation League is a competition where agents work in a simulated disaster zone, controlling and coordinating disaster response teams to attempt to rescue civilians and preserve the city. Each team of agents plays a fixed set of maps, one team at a time, and each team is scored based on the number of civilians they save, or in cases of a tie, how much of the city they preserve.

The Simulation League began in 2001, with two competitions leading up to the World Cup in Seattle. In this competition, a Japanese competitor YabAI took top prize, defeating all other competitors, including the formidable competition from Iranian competitor Arian.

YabAI demonstrated several weaknesses in the competition in the fact that it won, since it used very little coordination. Many of the successes of YabAI stemmed from the emergent behaviour that resulted from the intelligence of each individual agent, and demonstrated the power of such effects in a disaster situation.

1 Introduction

RoboCup Rescue is a competition designed to inspire research into coordination and technology that can be used to improve responses to natural disasters, such as earthquakes and tornadoes. This is divided into two leagues: the *robot league*[1] focuses on robotic technology, where an operator must remotely command a robot to locate injured victims in a debris field. The *simulation league* focuses on coordination of virtual 'platoon agents' after an earthquake in a simulated disaster zone.

2 RoboCup-Rescue Simulation League 2001

The 2001 RoboCup-Rescue Simulation league consisted of two open competitions in Fukuoka and Osaka, Japan, and a 'World Cup' in Seattle, Washington. The final had seven competitors, of which YabAI placed first in the finals.

The simulation is a one-tenth recreation of the city of Kobe, over 500m x 500m area. The simulation models an earthquake at the beginning of the simulation, and five virtual hours following. A distributed set of software agents control the progression of the simulation, some of these being part of the simulation itself, and the rest being controlled by the current competitor.

The goal of the competitor agent teams are to save injured civilians and to prevent the destruction of the city by fire.

2.1 Architecture

The simulation is enacted by the *RoboCupRescue Simulation System*. The system is agent-based, to distribute the compute load of the simulation across several computers. A central *kernel* coordinates the control flow of the simulation and manages communication with each of the agents. The *Geographical Information System* connects to the kernel to provide details on the disaster space, including the map and details of the disasters that are going to happen. [2]

The *sub-simulators* update the state of the world to account with respect to each disaster. There is one agent for each disaster, for example, one causes buildings to collapse and marks debris on the roadways, while another causes fires to expand and ignite other buildings when not extinguished.

Viewers can connect to the simulator to display or analyze the simulation as it unfolds. A great deal of work has gone into this area, resulting in a myriad of different viewers being produced.

Finally, a general class of agents called the RoboCupRescue (RCR) agents are the actual interacting entities inside the simulation. This general class of agent is used to model civilians, as well as all the platoon and center agents competing in the competition.

2.2 Competitors

Each competitor designs a set of RCR agents. Each competitor agent represents either a single ‘platoon’ agent, or a single ‘center’ agent. Platoon agents are mobile rescue services, including fire brigades, ambulance teams, and police forces. Center agents are special agents that can be used to coordinate a matching class of platoon agents.

Each platoon agent has the ability to communicate with other agents in the immediate area, or globally with agents of the same type. They may each also sense the state of their immediate surroundings (30m). Each platoon type also has a special ability; ambulances may rescue buried civilians, police may clear blocked roads, and fire brigades can extinguish burning fires.

For the 2001 game, a team was limited to ten fire brigades, ten police forces, five ambulance teams, and one of each center. A competitor was required to implement the fire brigades at a minimum, and could use sample code for the other two platoon agent types. If a competitor desired a center agent, it had to be implemented by the competitor.

2.3 Simulation Agents

Seventy-two civilian agents are added as RCR agents to the system. These agents wander around the city and can be injured or killed by collapsing buildings or fire. The key purpose of ambulance teams is to search for and rescue these civilians, and then bring them to the hospital.

Two varieties of civilian agent exist. One does not communicate except when injured and buried, in which case it communicates "*HELP*" to the immediate vicinity. Others are able to either indicate that they themselves need help, or may also tell platoon agents of the approximate location of another agent they heard that needed help, or of road obstructions they have encountered. [3]

3 Issues

3.1 Communication-constrained Coordination

One of the most important goals of the simulation league is examining the coordination of multiple platoon agents. Communication is extremely expensive, costing a turn to send an outgoing communication. Inbound communication is limited to hearing four messages per communication medium per turn. Coordination across agent types is even more difficult, requiring the agents to be in close proximity to each other, or use of center agents which introduces a multi-turn communication latency.

A successful team must use communication effectively to coordinate agent activities, but must use communication sparingly to avoid congesting the communication medium or sacrificing too many turns of productive work.

3.2 Real-Time and Dynamic Planning

The simulation is real-time, with each agent given approximately two seconds to calculate its turn. If a deadline is missed, the agent misses the opportunity to act that turn. Agents must be kept relatively simple, or must use algorithms which can return a "best so far" answer at any time.

Due to the large numbers of unknowns, and the ability of game state to change over time, an agent must use dynamic planning. An agent must be able to account for unforeseen circumstances, and have the ability to re-plan in these cases. Prioritization should be used to allow an agent to pursue more important objectives, but an agent should also be able to commit to an objective to prevent it from constantly switching tasks.

3.3 Emergent Behaviour

Since the communication costs make fully central control impractical, each platoon agent must be able to act semi-autonomously. However, when multiple agents each act by their own initiative, the combined effect may be desireable, or may be counter-productive.

As an example, imagine a simple algorithm for ambulance building search that does not take other agents into account. If two ambulances ever co-operate to rescue a person, these ambulances could then end up moving in lock-step, essentially wasting the search effort of one of them.

As a positive example, imagine a fire brigade algorithm that only allows one agent to work on a specific building. Firetrucks would congregate on the highest priority fire, but then start working on adjacent buildings, potentially halting the spread of fire in other directions.

An understanding of how individual decisions will effect the emergent behaviour of the system is critical in constructing an effective competitor.

4 YabAI

YabAI was a competitor developed at the University of Electro-Communications in Japan. The competitor was written using Java, and its implementation later led to the development of YabAPI[5], a generic Java framework on top of which one can develop agents. A custom-written predicate and set logic package is provided, with much of the agent's intelligence being represented with basic set logic.

Details of the competitor are difficult to obtain, since only one (short and vague) paper[4] was written about it, and the source code is not available. Source code for a later revision[6] is analyzed for details where not available in literature.

4.1 Fire Strategy

No coordination is performed between fire brigades. Rather, YabAI prioritizes fires using the global knowledge of fires in the city. The agents then attempt to extinguish the highest-priority fires that it believes it can reach.

Sets are built using the predicate toolkit based on a fire's urgency. This roughly maps to what is surrounding the fire, with fires next to unburned buildings considered more urgent. This is divided into three sets (very urgent, urgent, and normal). Fires are also classified by their age, also divided into three sets (very early, early, and normal).

Fires are then ranked into fourteen categories based these six sets, and one extra flag, namely the closeness of the fire. This extra set ensures that fires within the immediate extinguishing range of the fire brigade get priority. A fire brigade will randomly choose a fire from the set of fires with the highest rank if it can reach the fire.

4.2 Rescue Strategy

Despite assertions to the contrary in their paper, there is no communication used to coordinate ambulance teams. Ambulances prioritize their actions, preferring to load and unload when possible, and then prioritizing the known injured citizens into four classes (need immediately, need, no need, and abandon). This information is not communicated to other ambulances; however, when there is no other known civilians, ambulances will use their knowledge of other ambulance positions to join another motionless ambulance. Motionless ambulances are likely in the process of freeing a trapped civilian, so this allows ambulances to cooperate without a communication penalty.

4.3 Road-Clearing Strategy

Police agents' sole purpose is to clear roads. They do not coordinate among themselves to do so, rather each officer reaching the first blockage recorded on their internal list, then moving on to the next blockage.

First, police officers concentrate on clearing roads reported blocked by other agents. These get priority, since other agents likely encountered the blockage in attempting to reach its destination. After this, it clears other blockages it knows of, and then finally choosing to explore roads it has yet to traverse to search for new blockages.

While they describe a system of clearing towards fire brigades or to fires to ensure clear paths, such does not seem to be the current state of the implementation.

4.4 Inter-Platoon Coordination

There is only one real use of communication for coordination: agents that encounter road blockages tell this on their broadcast medium. Police agents receive this and use it to build a list of blocked roads to clear. Police agents will also tell by broadcast when they have cleared a blockage so that other agents can incorporate this knowledge into their world models.

The center agents are little more than dumb relays, blindly repeating any message they receive. In fact, the 2001 score rankings[7] do not even acknowledge the use of center agents by the team at all!

There is no selective hearing of messages. The first four messages are blindly selected to be heard, with all other messages lost.

5 Results

In the final rankings[7] of the 2001 RoboCupRescue Simulation League World Championship in Seattle, YabAI was the victor in a competition of seven competitors.

During the preliminary round, YabAI came second, barely defeated by Arian, a competitor from Sharif University of Technology in Iran. During the semi-finals, YabAI faced a competitor Rescue-ISI-JAIST, matching its casualty count, and faring slightly better in extinguishing fires and preventing injury.

Arian faced a competitor JaistR, which won based solely on the number of casualties it prevented in two of three games. However, it was found that JaistR had broken the 'gentlemen's rules'[1] of the game, and the agent had to be disqualified.

During the final match, YabAI won both games played versus Arian, winning the first on civilian casualties, and the second on the number of injuries prevented and amount of city preserved.

6 Analysis of YabAI

The success of YabAI in this competition demonstrates the ultimate failure of the simulation league to achieve their objectives in this competition. The point of the simulation league is to model the communications and coordination of what would eventually become human agents with assistive devices. The final winner turned out to be one of the most primitive, being the one with practically no communication or coordination present!

YabAI's success can be attributed, other than just having weak competition, to the phe-

[1]Due to the unusually primitive nature of the simulator, the responses provided by agents are not verified when received. As such, agents must self-limit their behaviour to match game rules, and if it is observed that they do not, they are disqualified by the human judges.

nomenon of emergent behaviour. Each agent in YabAI is given the ability to operate autonomously, and their strength is not in coordination, but in the fact that each performs their task efficiently in isolation. With the small amount of global state available, the emergent behaviour gives the appearance of cooperation.

6.1 Areas for Improvement

YabAI could be improved in countless ways, most based to at least a small extent on actually making use of communication.

Center agents are vastly underused in YabAI. They are a powerful resource in that they have communication ability, as well as being an extra compute resource with no other task but to communicate. In later competitions, they are even given greater communication ability, to emphasize their importance to the competition.

Center agents could combine the information they hear into one informative message delivered to their agents. While one is supposed to send only one ‘sentence’ per message, the message could still contain a simple list of items rather than a message per item.

As an example, currently, the ambulance teams report road blockages to the ambulance center agent. Rather than relay both, it could simply combine the two into a message listing the blockages. In a more advanced version, the center agent could even make this message double as a form of coordination. In the previous example, the police center agent, rather than just sending two blockage messages, could send a message listing not only the blockages, but also which police agents should move to which.

Since the fire station will hear all messages regarding blockages, sees all fires, and knows the location of all agents, this seems like an ideal place for centralized control. Emergent behaviour saves on computational cost, but since the fire station and its corresponding broadcast medium are not really used, there is no reason not to add some basic central coordination. The fire station center agent could spend time calculating an optimal extinguishment strategy, possibly over several turns, and then send a message directing the fire brigades to redirect their focus on particular fires.

While seeming not to be a problem in the current communication setting, some basic intelligence should be added to the hear command, to actually have agents intelligently choose which messages to read. Currently, the first four messages are read without consideration of the recipient, which could become problematic. At very least, it would be more sensible to favour messages from center agents over fellow platoons. Also it would be wasteful to hear messages from the same civilian agent more than once, and should be secondary to coordination messages.

Listening to civilian agents could also be useful. Many of them reveal information about other injured civilians or road blockages, allowing the platoon encountering it to update its world model

Due to the high weight of civilian casualties, police agents could be redirected to search for injured civilians. Combined with basic central coordination, agents not near to any road blockages could be used to search buildings, especially those near fires that have not yet been searched.

7 Analysis of RoboCup Rescue

The success of a competitor like YabAI demonstrates several fundamental weaknesses in the RoboCupRescue Simulation League.

First off, the goal of the RoboCupRescue initiative is unclear. Focus seems to be scattered, with the Simulation League and mobile devices projects concentrating, supposedly, on coordination of human disaster support teams. However, many of the design issues in the Simulation League follow more closely that of the Robot League, where the disaster support teams themselves need to be developed as well. Finally, there is a blurred line between the desire to realistically simulate a disaster situation as it would actually unfold versus the desire to create agents to counter it.

Several issues do not mesh with reality. First off, their communications model needs to be much more specific and accurate. Assuming the communication is human-to-human is pointlessly inefficient when the entire point is to design a software-based coordination system. Being able to choose recipients in a saturated broadcast medium is equally unlikely, whether computer or humans are communicating.

The scoring function used to evaluate agents is ridiculous. They state that their guiding principle is “Human life is the most valuable.” [3] By their objective function, a single casualty results in a worse score than the entire city being burnt to the ground and all civilians and rescue agents being critically injured but not dead. While this may give the competitors a warm fuzzy feeling for saving virtual lives, it makes it impossible to effectively evaluate the true performance of the agents. Furthermore, the fact that the buildings and health values are multiplied makes no sense, as a good score in one will entirely cancel out a bad score in the other.

Several reports also indicate that the simulator was weak, with bugs and crashes occurring during the competition. The fact that the ‘gentlemen’s rules’ mandating that agents self-limit their actions had to exist at all further supports this analysis.

8 Conclusion

RoboCup Rescue was a good start at generating interest in an ever-expanding range of research problems for disaster response. The Simulation League in particular generated a great deal of interest in the RoboCup Rescue project, which later resulted in work on everything from PDA devices to disaster simulators.

What the Simulation League holds in goals and desires, it lacks in implementation. The winning competitor was far too weak to draw many conclusions on coordination from. The biggest lesson learned from YabAI was the importance of emergent behaviour in cooperative agent systems. This may be in part an issue of the competitors, but also in the design of the simulation itself.

With work and improvement over the years, RoboCup Rescue Simulation League will drive cooperative agent development, and will most certainly accomplish many of the project’s goals.

References

[1] “Robot League,” *RoboCup 2003 RoboCup Rescue,* April 16, 2003; http://www.isd.mel.nist.gov/RoboCup2003/ (current June 2004),.

[2] T. Morimoto, *How to Develop a RoboCupRescue Agent.*

[3] "Semi-Final Rule and Evaluation for 2001 RoboCup-Rescue Simulation Leagues," *RoboCup-Rescue Simulation Project,* April 1, 2001; http://www.rescuesystem.org/robocuprescue/rule2001.html (current June 2004).

[4] T. Morimoto, K. Kono, and I. Takeuchi, "YabAI The first Rescue Simulation League Champion," *RoboCup 2001,* pp. 44-59.

[5] T. Morimoto, "YabAPI: API to develop a RoboCupRescue Agent in Java," *RoboCupRescue;* http://ne.cs.uec.ac.jp/~morimoto/rescue/yabapi/yabai-0_50.tar.gz (current June 2004).

[6] "YabAI;" http://ne.cs.uec.ac.jp/~morimoto/rescue/yabai/ (current June 2004).

[7] "RoboCup-Rescue Simulation League Competition Report," *RoboCup 2001 Seattle;* http://www.rescuesystem.org/robocuprescue/robocup2001report.html (current June 2004).