18th World IMACS / MODSIM Congress, Cairns, Australia 13-17 July 2009
http://mssanz.org.au/modsim09

# Multi-Agent Coordination and Optimisation in the RoboCup Rescue Project

**Wing Yan Janice Chou[1], Luke Marsh[2] and Don Gossink[2]**

*[1]Adelaide University, Adelaide, South Australia, Australia*
*[2]Defence Science and Technology Organisation, Edinburgh, South Australia, Australia*
*E-Mail:* Luke.Marsh@dsto.defence.gov.au

**Abstract:** The RoboCup Rescue (RCR) simulation project is an open resource for research in the area of multi-agent systems. The domain is based on a scenario with a city undergoing certain catastrophes. The goal is to coordinate and control the emergency services in the city to minimize damage and injuries resulting from the disasters. The emergency services are represented by police, fire and ambulance software agents. The domain incorporates general command and control issues and concepts.

In this paper we propose a decentralised multi-agent architecture which is applied to the RCR simulation. The architecture is designed to coordinate the emergency services in the simulation. The multi-agent architecture is implemented via a communication protocol, and is based on the Contract Net Protocol (CNP), which operates similarly to a first-price sealed-bid auction system. Multi-agent interaction can occur in a many-to-many setting; with many customer agents interacting with many supplier agents at the same time, with the goal of allocating tasks to the best suited agent.

Agent decision making was managed using an agent control loop, shown in Figure 1, which was designed to control and synchronize each agents' actions. Optimal path planning within the RCR city is done using a modified A* search algorithm, called Online A*, which finds an optimal path in a network with changing path costs. Optimisation techniques were also used to improve the agents' performances in the scenario by prioritising the tasks to be completed, assigning an efficient number of agents to perform each task, and to determine which specific agents should complete each task. Experiments were conducted to validate the approach taken. The experimental results suggest that the decentralised agent architecture may be promising, but future research is required.

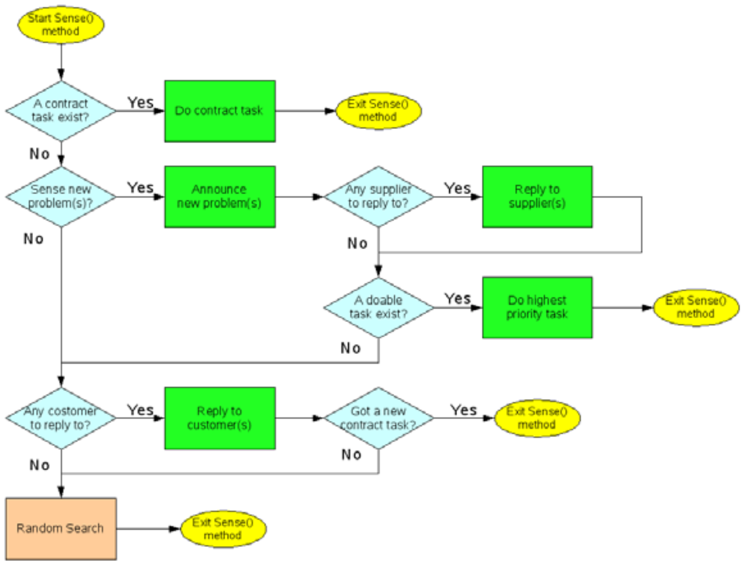

**Figure 1.** The agent control loop

***Keywords:*** *Multi-agent, Optimisation, Decentralised Architecture*

## 1. INTRODUCTION

During the recovery from incidents of terrorism and natural disasters, there are many issues in the coordination of different government and non-government organisations in responding to these emergencies. For example, responses for some incidents are duplicated while for others they are absent. Effective coordination among organisations (police, fire fighters, ambulance, military, etc.) for emergency response is critical to minimise the impact of the incident. Research into appropriate coordination protocols is necessary to achieve this goal.

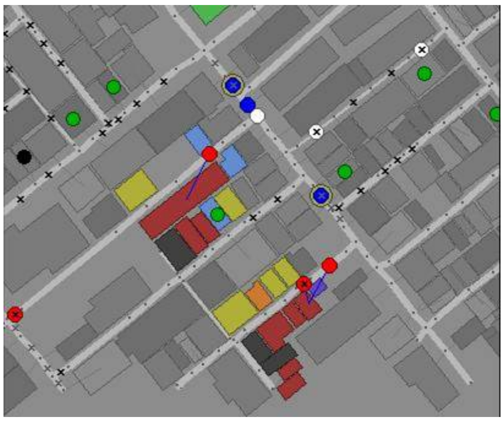

**Figure 2.** RCR simulation shows a top view of a city map with buildings (*rectangles*) and agents (*circles*).

In this paper we present a decentralized coordination approach, which is applied to the RoboCup Rescue (RCR) simulation environment by implementing a variant of the Contract Net Protocol (CNP) (Smith, 1980). An agent control loop was developed to direct agent decision making in the RCR simulation. Optimisation techniques were also used to improve the agents' performances in the simulation by prioritising the tasks to be completed, and efficiently assigning the agents to tasks. The simulation was executed and analysed to investigate the effectiveness of the agent architecture for coordinating organizations in emergency response.

## 2. OUTLINE OF ROBOCUP RESCUE SIMULATION

Our architecture is built on the RoboCup Rescue Simulation Project (Kitano and Tadokoro, 2001) (http://www.robocuprescue.org, Accessed 12 January 2009). The main goal of the RoboCup Rescue project is to promote research and development in this socially significant domain at various levels involving multi-agent team work coordination (Wooldridge, 2006).

In the RCR simulation, software agents that represent ambulance teams, police forces and fire brigades (rescue agents), interact in a software city model. Soon after a disaster, buildings may collapse, which cause civilians to be buried, fires could spread across buildings, and these cause great difficulties for the rescue teams to reach their target locations because of blocked roads. The main objective of the rescue agents is to minimize damage and injuries resulting from disasters. Figure 2 shows the simulation window of a particular city map. Rectangles represent buildings on the city; they are colour coded to show the degree of burning. Round dots represent rescue agents.

Table 1. Agent Actions

| Action Name | Description | Agent |
|---|---|---|
| move | Move to a specified location | All |
| rest | No action | All |
| extinguish | Extinguish a specified building | Fire Brigade |
| rescue | Rescue a trapped civilian | Ambulance |
| load | Load a civilian onto ambulance | Ambulance |
| unload | Load a civilian off ambulance | Ambulance |
| clear | Clear a specified road | Police |

In the RCR simulation, the three types of agents (Fire Brigade, Police and Ambulance) are controlled by three different multi-agent controllers that are programmed to match the agent's ability. At the start of the agent control, an agent is given a limited viewpoint of its surrounding environment, such as if any burning buildings and injured civilians exist. The agent can 'sense' a limited range of objects within a certain radius. With this data, the agent has to decide which action it should choose; this decision making will form the major part of the agent control loop. In general, agents can choose any actions in Table 1. All agents can use the move action; some actions are only applicable to a certain type of agent. The RCR simulation has been specified so that an agent can only choose one action for each simulation time step, and will use agent actions to change the simulated environment at each time step. This cycle continues for future simulation time steps.

Agents can communicate with each other by message passing; it is possible for an agent to send many messages in one time step. These messages can be sent and received by other agents almost instantly, even in one simulation time step. An agent can perceive a new task and request other agents to perform it, usually because it lacks the ability to perform the task. Agents who announce a new task and request responses from other agents can be referred to as customer agents. Agents who are interested in the new task, and who communicate back to the customer agents, can be referred to as supplier agents. In our RCR simulation, all agents will communicate using contract protocol methods, as explained in the following section.

## 3. MULTI-AGENT ARCHITECTURE

### 3.1. Multi-Agent Architecture

Our multi-agent architecture is implemented via a communication protocol that facilitates decentralized and deliberative planning and task allocation in this dynamic and open domain. The protocol acts similar to a first-price sealed-bid auction system. Agent interaction can occur in a many-to-many setting; with many customers interacting with many suppliers at the same time. We have chosen a decentralised architecture because unlike centralised architectures there is no single point of failure. In the RCR simulation, the agents are not in a competitive domain, they are cooperative, hence no conflicts arise, apart from competing for tasks to perform, which is resolved via the auction protocol outlined below. The agents also inherently cooperate, as they are all working towards the same goal, i.e. to rescue the city. Further cooperation is achieved by Police agents who free other agents that are stuck by blocked roads, and by Fire Brigade agents who can work together to put out fires. Some other work which has been done on multi-agent systems in RCR includes (Habibi et al., 2005), who also used an auction system for decentralised task allocation amongst their fire brigade agents, and Farinelli et al., (2003), have designed a multi-agent architecture which provides with planning, cooperation and information fusion capabilities.

The Contract Net Protocol (CNP) (Smith, 1980) is an approach to negotiation in multi-agent systems. CNP operates similarly to a first-price sealed-bid auction (Sandholm, 2000); where a customer agent announces a task to many supplier agents. All the bidders either submit bids to achieve the complete task or refuse to submit bids. After some deadline, the customer agent completes the contract by selecting the best bid according to its criteria. There are many protocols that are an extension and improvement over CNP, giving agents more options and alternatives in contract formation; some of these are Extended Contract Net Protocol (ECNP) (Fischer and Kuhn, 1993), CNP extension (Arknine et al., 2004) and Provisional Agreement Protocol (PAP) (Perugini, 2006). CNP and CNP extension require a supplier to have the ability to fully complete an announced task. ECNP and PAP allow partial bids. CNP extension, ECNP and PAP allow a supplier to negotiate with many customers at the same time. The protocol used in this architecture is a variant of CNP extension where it minimises the number of messages passed during the negotiation; this is similar to PAP, as PAP doesn't require the customer to reply to every supplier who had sent a proposal.

In our protocol when a customer agent announces a task, supplier agents need to make a decision on whether it can achieve the task, when it can achieve it and most importantly, whether this task is of high priority. Similarly when a customer agent receives many supplier proposals, it needs to make a decision on which agents can better achieve this task and what is a sufficient number of agents required for this task. If a contracted task is formally agreed by a customer and a supplier by using agent communication methods, this task is known as a contracted task; this is distinguished from a normal task that an agent might achieve by itself. This process takes four time steps: task-announce, proposal to do task, invite to do task, and offer (accept) task, as outlined below. If a supplier fails to win a bid, it will remember this loss and will not bid on this task again. The supplier will then choose the next highest priority task to bid on in the next time step, if the supplier has no other actions to perform.

A software agent in the RCR simulation can act as a customer agent who requests tasks or act as a supplier agent who performs tasks. The agent can take the role of a customer or supplier at different times during the simulation and it depends on the agent(s) it is communicating with. The following is a detailed explanation of some agent communication methods that are frequently used in the RCR simulation.

***Task-Announcement (Customer to Suppliers)*** The protocol is initiated when an agent wishes to achieve a task T; it acts as a customer and sends a task-announcement to contract out the task to all suppliers. The task-announcement is programmed to send to all the rescue agents (except the sender) in the RCR simulation. The task T can be one of the three tasks (extinguish a burning building, rescue an injured civilian or clear a blocked road). The task T includes a unique ID of the problem (e.g. the ID of a building), a proposed deadline and the location of the problem so that suppliers can travel to that location. Because it is possible for an agent to send many task-announcements during the simulation, a unique ID is attached to this task-announcement for easy identification.

***Proposal (Supplier to Customer)*** When an agent receives a task-announcement, it will act as a supplier agent. Only suppliers who can perform the task will send back a proposal; that is, only Fire Brigade agents respond to fire tasks, Ambulance agents respond to civilian tasks and Police agents respond to road tasks. When the supplier is not busy with other tasks, it can send a proposal back to the customer to indicate its interest in performing the task. (An agent can only send one proposal for a particular task.) The supplier will calculate the distance from its current position to the task location; this distance value will be included in the proposal which is required by the customer. If the supplier is responding to a fire task (i.e. it is a Fire Brigade agent), it will also include its water quantity value in the proposal. The distance and water quantity values are the criteria for which the customer agent uses to determine the best proposal.

***Invitation-to-Offer (Customer to Supplier)*** After the proposing deadline, the customer may have received some proposals to its task. It will decide on the required number of suppliers and choose the best supplier(s) to contract out the task. The customer sends an invitation-to-offer to the potential supplier agent(s).

***Offer (Supplier to Customer)*** When the supplier has decided to do the task, it submits an offer back to the customer; this formally makes the proposal an offer, and the supplier is committed to the offer. From this point onwards, the supplier can begin performing the task.

***Revoke-Offer (Supplier to Customer)*** If a supplier has offered to perform a task but afterwards cannot perform it for any reasons, it would send a revoke-offer to the customer informing its inability to complete the task. The supplier formally terminates the contract and is free to perform other tasks.

## 3.2. Agent Control

Figure 1 shows the program flow of the agent control loop. Agents are not required to run through every statement in the method; as there are several 'exit points', these are shown as ovals. A diamond shape represents a decision block. An agent's first priority is to complete a contracted task, if it exists. An agent may receive new contracted tasks through communications with customer agents. An agent may contract out a task through communications with supplier agents. The program flow is designed this way so that agents will keep themselves busy by finding tasks and doing them. Only when an agent has nothing to do will it reply to customers' messages. An agent will exit the agent control when it is performing a task (contracted task or highest priority task) or just received a new contracted task to perform. A rectangle represents a major decision and action of an agent. These major sections of the program are explained below.

***Do contracted task.*** If an agent has a contracted task then the program will enter the 'Do contracted task' section of the agent control. First the program will determine if the agent is at the location of the problem, if not the agent will need to move to that location. A modified version of the A* search algorithm (Nilsson, 1980), Online A* (Marsh et al., 2005), is used to derive an optimal path for the agent to travel. If the agent is at the right location, the program will decide on a relevant action for the agent to perform the task. Ambulance and Police agents must travel to the exact location of the problem for their actions to be valid; only Fire Brigade agents have the ability to extinguish fires when they are within a certain radius of the burning building. If a Fire Brigade agent cannot extinguish fires because its water tank is empty, the program will need to direct the agent to a refuge building to refill its water tank. If an injured civilian is found to be trapped, an Ambulance agent will need to first rescue it before the agent can load it onto its ambulance vehicle. Police agents may need to clear blocked roads along its path in order to reach the destination. Most of the actions may take several time steps to complete; this section of the agent control would be executed repeatedly until the task is performed fully. Finally the agent will remove the contracted task, and exit out of the agent control. If for some reason an agent cannot complete a task, the agent is required to inform its customer of the failure to complete the task. This could happen when a Fire Brigade agent cannot reach the burning building because too many roads along the path are blocked. In fact this happens quite frequently as there are many blocked roads at the start of the simulation. By sending this failure message to the customer, the supplier agent has freed itself from the contract and can remove it.

***Announce new problem*** When an agent does not have a contracted task to do, the program will determine if the agent has sensed any new problems. These new problems will be announced to other agents using the task-announcement method. For example a Police agent will task announce any new burning buildings, injured civilians and blocked roads when it senses them. The idea is that an agent who first senses the problem might not be the best agent to perform the task; so by announcing the task to everyone it is inviting agents who maybe better equipped for the task to deal with it instead.

***Reply to supplier*** If an agent has previously made a task-announcement and it received a supplier's message, then it may choose to reply back using agent communication methods. There are two types of supplier messages, proposals and offers, as mentioned in the previous section. If the supplier's message is a proposal, it is likely the customer would receive other suppliers' proposals as well. The customer will need to determine how many agents are required for the task and which agents are best suited for the task. To aid this decision making, each proposal includes a distance value (distance calculated from the supplier's current location to the task location), so the customer can easily sort the list of suppliers to find the closest supplier(s). If the supplier is a Fire Brigade agent, its proposals will also include a water quantity value. So the customer will need to take both suppliers' distances and water quantities into consideration when determining the best suppliers. If the supplier's reply is an offer, the customer is given a choice to accept the offer. If the customer accepts the offer, it would expect the supplier to perform the task fully.

***Reply to customer*** If an agent has a message from a customer agent such as a task-announcement or invitation-to-offer, it may choose to reply back using agent communication methods. If the message is a task-announcement, and the supplier agent has the ability to perform the task, then it can send a proposal back to the customer to show its interest. If an agent does not have the ability to perform a task (e.g. a Fire Brigade agent cannot clear a blocked road or rescue a civilian), the agent will just store the task in memory but ignore it. This is to prevent agents duplicating tasks as an agent is programmed to check if a task is announced previously before broadcasting the task announcement.

***Do highest priority task*** When an agent is not busy with a contracted task, it will try to perform a high priority task it has sensed. The definition of task priority will be explained in the optimisation section. This type of task is not setup using agent communication methods, but solely managed by each agent individually. The idea behind this is that even if an agent has broadcast a task out, because it has sensed it first, it probably is the closest agent to the task. So instead of waiting for other agents to reply back, perform the task, as it is highly likely this agent will be the best agent suited for the task anyway.

***Random Search*** When the agent has no other tasks to do, the agent will pick a random location in the city map and travel to that location, with the aim of sensing new tasks.

## 3.3. Optimisation

The number of rescue agents remains constant throughout the simulation. Optimisation in the simulation involves prioritising the tasks to be completed, assigning an efficient number of agents to perform each task, and to determine which specific agents should complete each task. In general, a problem (burning building, injured civilian or blocked road) in the simulation has a degree of severity. A burning building is measured in its fieriness level, which will slowly increase in time when they aren't being extinguished. A blocked road is measured in blockage levels; this value is set at the start of the simulation, and a higher blockage level equates to a longer time to clear the blockage. An injured civilian is measured in its health point level; this value gradually decreases over time unless it is rescued by an Ambulance agent. Given knowledge of these problems and constraints, a customer is programmed to choose the closest and the most competent suppliers. Only for building fire tasks are multiple agents potentially assigned, as outlined below. Each customer agent will choose the highest priority task to broadcast, based on the current fieriness level, health point level, or blockage level. Similarly each supplier agent will choose the highest priority task to bid for, based on the current fieriness level, health point level, or blockage level. Therefore priorities of tasks can change over time, however once the bidding process begins, the priority remains constant.

***Road Blockage Priority*** We have identified two types of road blockages. There is a type of blockage that is responsible for trapping a rescue agent and preventing it from leaving its location; another type is a normal blockage that all agents can sense. The first type of road blockage will be known as *AgentBlock* and the second will be known as *RoadBlock*. A rescue agent will discover it is trapped when it failed to search for a valid path to leave the area; it can send a task announcement to inform other agents of this type of blockage. For Police agents, an *AgentBlock* task has a higher priority than a simple *RoadBlock*; that is Police agents will always agree to perform *AgentBlock* tasks before *RoadBlock* tasks. As all Police agents have the same level of effectiveness in clearing roads, a customer agent will choose the bid from the closest Police agent to the road block to perform the task, and only one Police agent will be assigned to each task.

***Civilian Priority*** An Ambulance agent can rescue and transport one civilian at a time. Once it has loaded one civilian it needs to return the civilian to the hospital before it can load more civilians. As all Ambulance agents have the same level of effectiveness in rescuing civilians, a customer agent will choose the bid from the closest Ambulance agent to perform the task. Only one Ambulance agent will be assigned to each civilian task.

***Building Fire Priority*** There are five levels of fieriness in RCR that indicates the level of burning in a building; Not Burnt, Heating, Burning, Inferno and Burnt Out. A fire will usually spread to nearby buildings when its fieriness level is advanced to burning or inferno. When an inferno fire has been burning in a building for a long time, the building would eventually burn out and it is not possible to recover the building. A Fire Brigade agent is initially supplied with an amount of water in its water tank. If the agent uses the extinguish action, this water quantity will decrease an amount for each simulation time step. When an agent has used up all its water, it needs to refill its water tank and then return; this time delay seriously affects the success of completing the task, as the fire will cause more damage with time and possibly spread to other neighbouring buildings. There is a direct relationship between the amount of water required to put out a fire and a building's fieriness level; in fact higher levels of fieriness requires a Fire Brigade agent to refill its water tank at least once to fully extinguish a burning building. More Fire Brigade agents can be sent to simultaneously extinguish a fire to avoid agents having the need to refill their water tanks. A customer agent will choose the bids from the Fire Brigade suppliers who have enough water and who are the closest to the task location to perform its task. The number of agents required for a specific fire is directly related to the extinguishing time required to put out the fire as calculated below. The customer agent will then allocate the number of Fire Brigade agents as calculated below to each building fire task, if they are available. It is possible to calculate the number of time steps required to extinguish a burning building, given that the attributes of the building are known. Each building has a temperature value, when this value increases to the ignition temperature (denoted as *ignitionTemp*, which is set to 400), the building starts to burn. Therefore the goal is to reduce the buildings temperature to below *ignitionTemp*. The current *energy* and *ground-area* of a building are given by the simulation. The *capacity* of a building is a constant. The *effect* is a non-zero positive number and relates to how much a single Fire Brigade can reduce the temperature of a building per time step. Since each Fire Brigade uses a constant amount of water when extinguishing a fire each time step. The exact number of fire brigades can be calculated; to ensure that a fire will be fully extinguished before all of the Fire Brigade's are required to refill their tanks

$$capacity = \left(\frac{ground - area \times floors \times 3}{10}\right) \times 4 \quad (1)$$

$$currentTemp = \frac{energy}{capacity} \quad (2)$$

$$\Delta Ignition = currentTemp - ignitionTemp \quad (3)$$

$$timeSteps = \frac{\Delta Ignition}{effect} \quad (4)$$

## 3.4. Optimal Path

To complicate the movement in the simulation, agents are unable to reach a specified destination if the path along the way has a blocked road; agents are only able to travel up to the location of the blocked road and will remain stuck there. Since agents are given a limited view of its surrounding environment, they have no knowledge of road conditions in the entire city. An agent will remember the condition of a particular road (and store this in its memory) if its location is close to the road or it had been near the road previously. The architecture developed uses a modified A* search algorithm (Nilsson, 1980), called Online A* (Marsh et al., 2005), which finds an optimal path in a network with changing path costs. This search algorithm was chosen because it solves the exact routing problem inherent in the RCR simulation. It includes a list of road blockages as inputs, and considers these in its path calculations. During searching, if a road is blocked, Online A* will remove this road and its possible connections from the list of possible paths. Additionally, an agent can use other agents' knowledge of road conditions as inputs. When an agent notices a new road blockage that has not been announced before, it will use a task-announcement method to inform other agents about it. An agent doing a path search use this extra list of blockages to remove any reachable paths, therefore increasing the chance of finding a shortest and reachable path to its destination.

## 4. SIMULATION RESULTS

RCR has a scoring system that measures the level of chaos in the city. The score is mainly affected by the number of agents (rescue agents and civilians) alive and the fire conditions of all buildings in the city map, therefore the higher the score at the end of the simulation the better you have performed. At the start of the RCR simulation when all of the buildings are intact and all of the civilians are alive, the score is initialised; a disaster is then simulated, and as agents/civilians perish and buildings burn the score is reduced. The score is calculated at each time step using the following equation $(a + h / i) * \sqrt{n / t}$, where *a* is the current total civilians alive, *h* is the total health points of each alive civilian, *i* is the initial total health points of civilians at the start of the simulation, *n* is the current total non-burn building area, *t* is the total building area. To test the effectiveness of our decentralised agent architecture, trials were run using the default city map in RCR (Kobe), for this map the score is initialised to 97. Each time the simulation is run the agents, blockages, and civilians are placed at random locations on the map. The RCR simulation supplies single centralised heuristic and dummy agents, collections of these single agents were used as a benchmark to test the effectiveness of the decentralised multi-agent architecture. In total three sets of one hundred simulations were run, each with a five minute duration, with the current score recorded at twenty-five second intervals. Figure 3 shows the average recorded scores for the simulations at twenty-five second intervals, the line with the circles represents the scores obtained using the developed agent architecture, the line with the squares represents the scores obtained using the dummy agents in the RCR simulation, and the line with the triangles represents the scores obtained using the heuristic agents in the RCR simulation. For the agent architecture, the score at time

300 is 54.44±4.25, for the heuristic agents, the score at time 300 is 47.57±4.91, for the dummy agents, the score at time 300 is 25.56±2.14. It can be seen that the decentralised multi-agent architecture outperformed the centrally controlled collection of dummy agents; however it is difficult to infer any comparisons between the multi-agent architecture and the centrally controlled collection of single heuristic agents. A possible reason for the large standard deviations is that in the map scenario used in the trials (Kobe), four key buildings begin the simulation on fire. As fires spread quickly to other buildings, to obtain a high score it is important to extinguish them quickly. For both the multi-agent system and the heuristic agents it highly depends on the initial decisions of the police agents, to what score is obtained. So in the case where the police agents quickly clear a path to the initial fires, the fire brigades are able to extinguish the fires quickly and a higher score is obtained. If the police agents fail to clear a path to the initial fires rapidly, the fires get a chance to spread quickly, and a lower score is obtained.

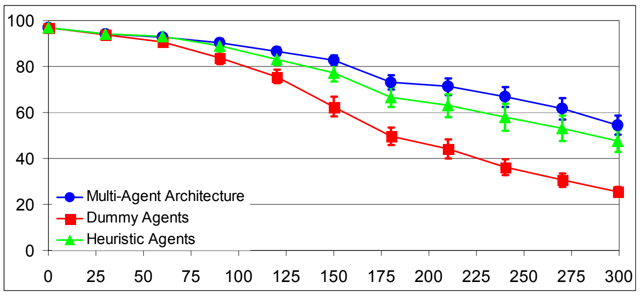

**Figure 3.** Simulation results, average score for 100 trials with standard deviation as error bars (y-axis), simulation time in seconds (x-axis).

## 5. CONCLUSION

This paper presented a decentralised agent architecture applied to the RoboCup Rescue project. The architecture was implemented using a messaging protocol operating similarly to a first-price sealed-bid auction. An agent control loop was created to manage agent decision making. Various optimisation methods were employed to improve the agents' performances in the simulation. Experiments were conducted to test the approach taken. The experimental results suggest that the decentralised agent architecture may be promising, but future research is required. For a decentralised system of multi-agents, a large amount of communication is required for coordination and task allocation. This is an important point when considering using a decentralized architecture in other command and control domains where bandwidth may be limited.

## 6. FUTURE WORK

Further work can be done on the decentralised multi-agent architecture to improve the performance of the emergency rescue and the score results in the simulation. At the time of writing, work on the optimisation modules of the agent architecture was not complete, further refinement is required.

## REFERENCES

Arknine, S., Pinson, S., and Shakun, M.F. (2004), An Extended Multi-Agent Negotiation Protocol. *Autonomous Agents and Multi-Agent Systems*, 8(1), 5-45.

Farinelli, A., Grisetti, G., Locchi, L., Lo Cascio, S., and Nardi, D. (2003), Design and evaluation of multi agent systems for rescue operations. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS03), Las Vegas, October 2003.

Fischer, K. and Kuhn, N. (1993), A DAI Approach to Modeling the Transportation Domain. DFKI Report: RR-93-25.

Habibi, J., Fathi, A., Hassanpour, S., Ghodsi, M., Sadjadi, B., Vaezi, H., Valipour, M., Impossibles Team Description. RoboCup 2005, Osaka, July 2005.

Kitano, H. and Tadokoro, S. (2001), RoboCup Rescue: A Grand Challenge for Multiagent and Intelligent Systems. *AI Magazine*, 22(1), 39-52.

Marsh, L., Calbert, G., Tu, J., Gossink, D., and Kwok, H. (2005), Multi-Agent UAV Path Planning. MSSANZ Conference on Modelling and Simulation (MODSIM 05), Melbourne, December 2005.

Nilsson, N. J. (1980), *Principles of Artificial Intelligence.* Tioga Publishing Company, 72-88.

Perugini, D. (2006), Agents for Logistics: A Provisional Agreement Approach. The University of Melbourne, Victoria, Australia.

Sandholm, T. (2000), Issues in Computational Vickrey Auctions. *International Journal of Electronic Commerce*, 4(3), 107-129.

Smith, R.G. (1980), The contract net protocol: high level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12), 1104-13.

Wooldridge, M. (2006), *An Introduction to MuiltiAgent Systems.* Wiley, 23-25.