RoboCup Rescue 2011
Rescue Simulation League Team Description
IAMRescue (United Kingdom)

Francesco Maria Delle Fave, Ramachandra Kota, Sam Miller,
Bing Shi, Sebastian Stein, Matteo Venanzi, and Nicholas R. Jennings

School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK

Abstract. In this paper, we outline the different software agents that are used by the IAMRescue team for the RoboCup Rescue Simulation League 2011. Specifically, we present a number of novel strategies that help our agents coordinate effectively within and across their respective ambulance, fire brigade and police force teams. For ambulance teams, we developed a dynamic scheduling mechanism for rescue operations. Furthermore, we designed police forces that strategically coordinate to remove blockades based on priority levels. Finally, we designed our fire brigades to contain fires within blocks of buildings and prevent them from spreading to the more vulnerable blocks where there are more civilians at risk. These strategies have been used successfully at RoboCup 2010 in Singapore, where the IAMRescue team achieved second place overall.

1 Introduction

Today, there is considerable endeavour in the domain of disaster management for distributed, agile and autonomous response in environments where uncertainty, scarcity of resources and bias are endemic. The RoboCup Rescue platform addresses some of these issues in this domain. In particular, the aim is to design effective heterogeneous agents that manage the behaviour of ambulances, fire brigades and police forces. In more detail, the challenges are as follows:

- To develop a distributed system architecture which operates effectively in the uncertain and dynamic environment of RoboCup Rescue where communication, and thus coordination from a central point, is constrained.
- To design agents which collectively plan to achieve common goals.
- To design effective behaviours that ensure desirable overall properties (as an emergent behaviour of agents' local decision-making based on incomplete, imperfect information).
- To devise techniques to allow agents to effectively balance acting and information sensing.

To address these challenges in the RoboCup Rescue domain, we developed the IAMRescue strategy, which defines the behaviour of a number of different types of agents, including the ambulance team, fire brigade and police force, as well as their respective centres, which are required to facilitate communication between agents. The novel contribution of our work lies in the main decision-making processes for each of our agents, which are based on sound, principled techniques and algorithms from the areas of artificial intelligence, distributed optimisation, search and scheduling. In drawing from these established fields and extending them, our team is not only able to perform effectively in the competition, but also applies and advances the state of the art of the research community in the context of a real-world problem.

Against this background, we now describe in detail how we design these different agents to address the challenges outlined previously for effective disaster management. We will concentrate on a number of main components:

– Decision-Making: the high-level strategic behaviour of the agents (Section 2).
– World Modelling: how we model the agents' view of the world (Section 3).
– Disaster Prediction: how the agents predict disasters (Section 4).
– Communication: how the agents coordinate with each other by communicating (given a limited communication capability) (Section 5).
– Routing: how the agents calculate valid paths to their destinations (Section 6).

In addition to these, we will discuss our overall software architecture (Section 7), previous experience (Section 8) and the improvements we have made since our last participation (Section 9).

2 Decision-making: Agent skills and action selection

In order to be able to encode the agents' strategy in a way that makes it possible to alter the team's aims and objectives at different levels (e.g. at the level of goals, plans, and actions), we decided to divide the decision-making processes into three principal layers which are responsible for making decisions at varying levels of abstraction:

– The strategic layer is concerned with making high-level decisions about which goals a larger team of agents should pursue. Such decisions include which burning buildings to extinguish, where to unblock roads, or which groups of civilians to rescue next.
– At the tactical layer, these high-level goals are translated into a specific plan for a set of agents, and hence a specific task for individual agents. For example, given the goal to extinguish a number of buildings on fire, this layer allocates fire brigades to the buildings, based on factors such as their location or intensity.
– The operational layer is finally responsible for realising an individual agent's task as a sequence of atomic actions. This includes low-level decisions, such as finding the optimal route to a destination, deciding where to aim water jets and when to re-fill water tanks.

This hierarchy has been used successfully in disaster-response applications to abstract and coordinate decision-making within multi-agent systems [1]. We employ it here for a number of reasons. First, it provides a natural way of building well separated and modular decision procedures, where the detailed realisation of high-level decisions is left to progressively lower levels (which, in turn, do not need be concerned with the reasoning to arrive at these decisions). Second, these layers translate well to the distributed nature of the RoboCup Rescue domain, where some team decisions can be delegated to a centre, but where most low-level decisions need to be taken by individual agents that have their own view of the world. However, in our current implementation, all three layers are typically executed by all platoon agents (which coordinate their strategic and tactical decisions using both explicit coordination messages and implicitly through their locations or other cues). Nevertheless, our current decision-making architecture supports delegating some of these decision to a centre in future work.

In the following sub-sections, we discuss in more detail the decision-making procedures that the different IAMRescue agents adopt.

2.1 Ambulance Team Strategy

The main task of the ambulance team agents is to determine how to best save victims trapped in collapsed buildings. To this end, we employ heuristics in the area of dynamic real-time scheduling. In more detail, we determine for each victim i a deadline $t_d^i \in \Re$ and the number of ambulances $N_{opt} \in [0, \infty]$ that are needed to save the civilian by the deadline (except if it is not going to die by the end of the game). Each of these victims (tasks), arrive in real time and may have changing deadlines due to external factors such as the building it is situated in catching fire.

To deal with this, we follow a distributed game-theoretic approach from [2] which allows for ambulance agents to make local decisions and consequently reduce the amount of communication required. In more detail, we approximate the allocation problem described above using a series of static *potential games*, and then use a decentralised method for solving the approximating games that employs the distributed stochastic algorithm (DSA). Our algorithm is run by each individual agent and proceeds as follows:

```
1. for all $i \in A$ do
2.   Estimate marginal contribution to the task of saving a particular civilian $t_d$
3.   Assign ambulance $i$ to civilian with highest marginal reward $t_d$.
4. end for
```

In estimating an agents marginal contribution to the task of saving a particular civilian, we use a trade off between travel time to the civilian and deadline to save the civilian. In doing so we can tailor our strategy during a particular disaster setting in order to increase the amount of civilians saved. Since the underlying problem is a potential game, a greedy approach in the local problem will improve the global solution. With such an approach, there are a number of issues that could lead to computing sub-optimal solutions (which we do not elaborate here due to space restrictions — for details, see [2]). Furthermore it relies on several estimates that are sometimes imprecise. To address these issues, we are currently working on several improvements to the ambulance strategy:

1. Perform a better allocation by computing the trade-offs in choosing an agent with an earlier deadline as compared with a number of agents with longer deadlines. This would avoid the agents wasting time to save one agent as opposed to working in parallel or together on other agents. In so doing, it would be possible to increase the number of agents that can be saved.
2. Improve multi-agent teamwork by coordinating across different teams. For example it would be possible for ambulance agents to coordinate with fire brigade agents by sending them the locations of the victims they aim to save and, in turn, fire brigade agents would protect these locations from fires that are spreading.
3. Reduce the need for communication by employing techniques such as [3] to allow agents to better solve the global problem on their own.

2.2 Fire Brigade Strategy

The purpose of the fire brigade agents is to put out fires. However, due to scarce resources (both limited numbers of agents and restricted water-carrying capacities), it is usually not possible to put out all fires. In such a case, it is vital to prioritise fires with respect to regions within the world, and if these fires cannot be put out, they need to be contained in order to minimise damage.

We begin by first defining a graph that models how fire can spread between buildings. Every node represents a building and every edge represents a heat transfer channel. By doing this, we model the dynamics of fire, and use this for the fire brigade strategy. Once the graph is built, we further define clusters of buildings. The rationale of dividing the world into clusters is that fires spread faster within clusters (given the proximity of buildings) rather than across clusters. Next, we define a fire site as a group of adjacent buildings that are on fire. Note that these buildings might be part of different clusters and might be separated by roads. Essentially, a fire site represents the extent of a fire as it spreads from its initial point. When two or more fire sites meet, they are combined into a single fire site. The perimeter of the fire site is essentially a set of buildings on fire on the perimeter of the fire site which threaten to burn adjacent buildings.

Against this background, we now detail how we prioritise clusters and fires from a fire brigade's perspective as ordered below:

- *Clusters:*
  1. The number of civilians and the criticality of their injuries are the foremost reasons for protecting a cluster first.
  2. Because fires spread in all directions, a fire site at the centre of the world is more dangerous than one on the perimeter of the world, since the fire is likely to spread to more clusters adjacent to it. Thus, the more central fire sites are, the more important they are.
  3. Clusters that are far from a fire site are less important.
- *Fire sites:*
  1. Fire sites with more clusters that are more at risk (as described above) are more important.
  2. Fire sites that are closer to the fire brigade are given more consideration.
  3. Fire sites that are more central (based on the same intuition as with clusters) are more dangerous and should be prioritised as such.
  4. Small fire sites are easier to put out (preventing them from spreading, such that they need not be contained). Thus, we prioritise small fire sites over larger ones which generally need to be contained and cannot be put out completely.

By giving different weights to these priorities, we are able to effectively decide first, which fire site will be more damaging in the future, (i.e. in a number of time steps, see Section 4 for more details of our disaster prediction techniques). Given this critical fire site, the aim is to contain the fire, preventing it from spreading to the more important clusters. Next, we consider the clusters that are adjacent or contain the perimeter of the critical fire site. By considering the priority of these clusters, we determine which is the most important cluster to protect. To effectively contain the fire from spreading to that cluster, we consider putting out the burning building on the perimeter that is closest to the critical cluster, defined as the critical fire-building.

Now, because of the local decision-making constrains of our agents, based on their individual beliefs of the world, they all locally determine a critical fire-building to put out. Specifically, at each time step, a fire brigade calculates a critical fire site, a critical cluster and finally a critical fire-building. Due to uncertainties in the world, they do not necessarily calculate the same critical building. Thus, there is usually an emergent coordination effort to extinguish these burning buildings, with some fire brigades often breaking off to focus on other (possibly closer) fire sites.

As future work, we intend to have more explicit coordination among fire brigades (with the fire centre coordinating the efforts) with a reasonable communication over-head. Furthermore, we intend to have more coordination with ambulances (with the ambulance centre coordinating with the fire centre). This would allow us to better prioritise clusters. In particular, ambulance efforts to save civilians in a cluster would increase the priority of that cluster. This would allow the fire brigades to slow down a fire sufficiently for ambulances to save civilians. Finally, more work will go into managing priorities of clusters and fire sites which essentially determine the behaviour of fire brigades.

2.3 Police Force Strategy

The aim of the police agents is to keep the roads clear from blockades, in order to allow the other agents to execute their strategies. Clearly, doing this quickly and efficiently is essential for the performance of the team, especially as most agents are initially highly restricted in their movement or even completely blocked. To address this, the strategy of the police agents includes two parts. Firstly, they ensure that all agents are part of a fully connected road network, and secondly, they clear important roads on the map, to allow faster access to strategic locations. We describe both parts in detail in the following.

Initially, the police agents give priority to ensuring that all agents have access to the refuges and to fire sites. To do this, the police team creates a list of goals that need to be completed:

– Each police agent must have a clear path to a refuge.
– Each other platoon agent must have a clear path to a refuge.
– Each fire site must have a clear path to a refuge.

The goals are processed in the order shown above. Once they are completed, every agent, fire site and refuge is then guaranteed to be on a fully connected road network, allowing all agents access to critical positions on the map. For efficiency, these goals are allocated to agents such that the overall time spent on clearing roads is minimised. In more detail, we estimate the time it would take each agent to complete each goal and then employ the well-known Hungarian algorithm to allocate agents to goals in an optimal manner. As most of the knowledge about the world state is shared through communication, this is done locally in a decentralised manner in order to save time and bandwidth.

Once all the above goals have been achieved, the police agents start clearing strategic roads on the map, in order to improve access to important locations. This is done by attaching a value to each blocked road, which depends on two factors: (1) whether the block is generally in a highly-visited location (e.g. near a refuge or along a main stretch of road) and (2) whether it is currently on the shortest path between a refuge and an object of interest (e.g. a civilian or a fire site). To reduce the computational burden for this strategy, police agents do not coordinate explicitly when selecting roads, but rather each agent adjusts the value of a road depending on its proximity and then greedily clears the most important road that is not already being cleared by another police agent. This ensures that agents generally clear different roads in their current vicinity.

This concludes the outline of our ambulance, fire brigade and police strategies. In the following sections, we turn towards more general decision-making procedures that are shared by most agents.

3 World Modelling

The world model of each agent represents a view of the current state of the world, including highly dynamic information about the conditions of buildings, the locations of civilians and the states of other agents. As such, it is an integral part of each IAMRescue agent and provides the basic input data for our decision-making procedures.

At the most basic level, our world model stores the direct observations that an agent makes each time step. However, as this is clearly limited to the immediate surroundings of each agent, we use radio messages to relay observations to other agents. Since communication is generally limited, we typically allocate teams to different communication channels and use the centres to forward information between the channels (see Section 5).

Now, in order to build and maintain a comprehensive world model, we explicitly include some monitoring and information gathering actions in the operational layer of each platoon agent (regardless of its type). These activities usually become active when no tasks of higher priority is available, or when a new simulation run has just started. In particular, at the beginning of a run, all agents start by searching the map. This means going into every building that is not known. In so doing, agents can make use of a number of cues available to them in order to enhance the search for a victim or fires. For example, if an agent hears a victim calling for help and if it has never been heard before, the agent needs to search all buildings that are within its hearing distance. Also, generally, agents will search the closest as-yet unsearched building, but when several agents are close to each other, they consider only buildings in a precomputed individual partition of the map. This is to avoid causing agents to select the same buildings to search.

4 Disaster Prediction and Parameter Learning

Disaster prediction and parameter learning are essential aspects of our different strategies. In particular, they allow us to tailor our behaviour based on past and future beliefs of the world. We now look at these two aspects in more detail. In the RoboCup Rescue platform, damage (to civilians, rescue agents and buildings) is caused by fires. We believe that knowledge of how fires spread is essential to coordinate behaviours of agents. Thus, in our fire brigade strategy, we predict how fires spread. Such knowledge of potential damage of a particular fire site is useful when deciding which fire site warrants a higher extinguishing effort. Now, because of the uncertainties in the world, hidden simulation parameters such as temperatures of buildings have to be inferred. This often results in more inaccuracy in predictions the further we look in the future. Thus, it is usually better to predict only a few (10 in our fire brigade strategy) time steps ahead. Our agent runs its own simulation of the world with buildings burning and fires spreading (based on its current view of the world) for a number of time steps and uses its future belief of the world when deciding on which fire site is more critical.

5 Agent Communication

Communication is vital in aiding the agents' strategies, because it ensures that all agents have a consistent and up-to-date world model. However, because communication bandwidths and listening capabilities for the agents are limited, we use an algorithm that flexibly divides agent teams onto the available channels in a fair manner, allocating multiple channels to each team where possible. While platoon agents communicate information only to their assigned team channels, centres, which can typically listen to more channels than platoon agents, are used to relay information between the respective teams. When these centres are unavailable (e.g., when they burn down or when the scenario contains no centers), platoon agents dynamically start to share their information on the other teams' channels. This approach ensures that the available bandwidth is maximised, that all teams receive a fair share, and that information is eventually propagated to other agents.

To deal with failures, agents repeat messages within the same time step when there is spare bandwidth on a channel, thus increasing the probability that at least one of these copies is received. Each agent also monitors its own channels and repeats messages that were sent but not heard on the next time step (as expected).

6 Routing

Routing is an essential part of the operational decision-making layer of each agent. As mentioned before, the strategies for the ambulances, fire brigades and police forces critically depend on efficient computation of routes between two points on the map, as well as the availability of accurate estimated travel times. In our current implementation we use Dijkstra's algorithm to find the optimal route agents should follow to reach their destinations. The main issue with the use of Dijkstra's algorithm is the computational complexity, which is $O(E + V \log V)$ in the worst case, where V is the number of nodes of the graph and E the number of edges. In the RoboCup Rescue domain such a computational complexity can be problematic, because maps can be very big and the computation should be repeated every time a road changes its state (becomes blocked or is freed by a police agent). However, a good property of the Dijkstra's algorithm is that the output of its computation is a routing map from all nodes to a given source. Finding an optimal path from all nodes to the source, on this routing map, is then very easy. Therefore we can compute the routing map only once at each execution cycle considering as the source the starting position of the agent.

Furthermore, several factors of uncertainty, mainly due to possible unconsidered blockades along the path, need to be taken into account for path planning. This is ongoing work that is currently being implemented for the IAMRescue team. To this end, we plan to use probabilistic estimates to assist the routing algorithm when there is no evidence of whether a road is clear or not. By combining geographical distance with the probability of encountering a blockade, the routing algorithm will be able to detect routes which are likely to have blockades, for example if they are close to potentially collapsed buildings.

7 Software Architecture

Figure 1 shows the overall modular software architecture we adopted for our software agents, highlighting the main components and the information flow between these and the agent's environment. Here, four auxiliary modules heavily support the agent's decision-making — the world model contains facts about the world (Section 3), the communication module encodes and decodes messages and ensures the world models between separate agents are consistent (Section 5), the routing module is responsible for finding the shortest path to a given destination (Section 6) and the execution module submits low-level commands to the RoboCup Rescue server. The agent's main logic then uses these components to implement the strategies described in Section 2.

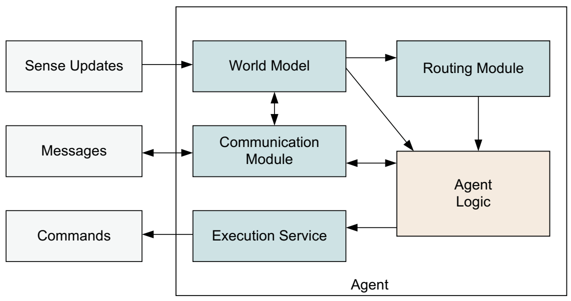

Fig. 1. Architecture of an IAMRescue agent

8 Experience and Performance

In this section, we briefly discuss the history of our team and past performance. IAMRescue has participated in a number of RoboCup tournaments, most recently in 2010, where it achieved second place overall, out of 12 initial teams. The success of our strategy was largely due to its ability to perform well in a wide range of settings, including those with no communication, failures on channels and missing channels. This was achieved by applying sound and general scientific approaches rather than hard-coding the agents' behaviours for specific scenarios. In doing so, we also built on work that has been rigorously evaluated in academic papers [2, 3].

Table 1 shows the results of the 2010 final, where IAMRescue became runner-up of the RoboCup Rescue Simulation League. As can be seen here, our team performed poorly in the initial two rounds of the day (coming last twice in succession). This was due to a software bug that had been introduced by accident during the night preceding the final and that we only managed to remove by the third round. As this significantly affected our performance during the final, we are this year

Team	Paris6	Berlin5	Paris7	Berlin6	Paris8	Total Score	Total Points	Final Rank
RoboAKUT	1	1	1	1	4	8	102.703	1
IAMRescue	4	4	2	2	1	13	83.844	2
ZJUBase	2	2	3	4	2	13	69.193	3
Ri-One	3	3	4	3	2	15	64.606	4

Table 1. Results of the RoboCup Rescue Final 2010

concentrating not only on improving our overall strategies, but also on improving the robustness of our code and developing a comprehensive test infrastructure. We summarise both in the following.

9 Improvements since 2010

Since our participation at RoboCup 2010, our team has been undergoing a number of significant improvements. These have largely been included in the preceding sections, but the following summarises these:

– We are concentrating this year on improving the *coordination* between the platoon agents. A key issue last year was that agents often pursued the same goals (based on greedy local decisions), especially in settings where communication was restricted. To address this, we are developing new decentralised mechanisms for better coordinating the task assignment between agents.
– We are extending the routing mechanism to take into consideration uncertainty regarding the presence of blockades on unknown roads. This causes agents to explicitly balance the utility of choosing a slightly longer, but certainly free road over a shorter but unknown road.
– We are optimising the division of channels to the different agent teams to better utilise the available bandwidth and achieve a fair share for all teams.

Additionally, as described in the previous section, we have this year developed a comprehensive test framework that will ensure our algorithms are more robust than last year. This is required because RoboCup Rescue is a project where many developers are involved in working on different parts of a modular software architecture. Consequently, extensive testing is absolutely necessary in order to validate and document the continuous changes affecting the code. For this purpose, we developed a test framework which is able to repeatedly run the IAMRescue team on different maps and settings. Results are automatically collected, and statistics are periodically calculated.

10 Conclusions

With our solid strategy that achieved second place in 2010, as well as the numerous improvements we have made over the last year, IAMRescue is well-positioned to not only represent a competitive entry at Robocup 2011, but also to advance the state of the art in multi-agent systems. We are able to achieve this by applying sound scientific approaches to the complex problem of disaster management, which we envisage to be transferable to a range of related problems and applications.

References

1. Siebra, C., Tate, A.: I-rescue: A coalition based system to support disaster relief operations. International Conference on Artificial Intelligence and Applications (AIA-2003) **101(1)** (2003) 289–294
2. Chapman, A., Micillo, R.A., Kota, R., Jennings, N.R.: Decentralised dynamic task allocation: A practical gametheoretic approach. In: The Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS '09). (2009) 915–922
3. Williamson, S.A., Gerding, E.H., Jennings, N.R.: Reward shaping for valuing communications during multi-agent coordination. In: The Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS '09). (2009) 641–648