# Decision and Coordination Strategies for RoboCup Rescue Agents

Maitreyi Nanjanath, Alexander J Erlandson, Sean Andrist, Aravind Ragipindi, Abdul A. Mohammed, Ankur S. Sharma, and Maria Gini

Department of Computer Science and Engineering, University of Minnesota
{nanjan,gini}@cs.umn.edu

**Abstract.** We describe the decision processes of agents in the Robocup Rescue Agent Simulation. Agents have to rescue civilians trapped in buildings and extinguish fires in a city which has been struck by an earthquake. Lack of information and limited communications hamper the rescue process. We examine how effective our strategies and algorithms are and compare their performance against the baseline agents and agents which competed in last year's competition.

## 1 Introduction

Disaster management and urban search and rescue is an open area of research for AI and multi-agent systems. This research not only has the potential for a huge social impact but also presents plenty of challenges. A search and rescue system has a large number of heterogeneous agents who have to act in real-time in a difficult environment with limited information while confronting problems of logistics, planning and collaboration.

The RoboCup Rescue Agent (RCR) simulation was proposed in [1] to advance research in the area of disaster management and urban search and rescue. It provides a platform for disaster management where heterogeneous field agents (police, fire brigades, and ambulances) co-ordinate with each other to deal with a simulated disaster scenario [2]. Police agents have to clear road blockades to provide access to the disaster sites, ambulance agents have to rescue civilians, and fire brigade agents have to control the spread of fire and extinguish it. The simulator also provides centers, a Police Office, a Fire Station, and an Ambulance Center, to help the field agents coordinate. Centers are critical for coordination, because the number and size of messages each agent can send or receive in each cycle are severely limited. The agents are given a map of the city but are unaware of the locations of civilians, blocked roads or burning buildings. They have to search the area to discover emergencies and coordinate with other agents.

Figure 1 shows a screenshot of the simulation environment. It is a portion of Kobe city where an earthquake of magnitude 6.9 on the Richter scale struck in 1995 damaging almost the entire city and causing a considerable loss of life and property. The blue, white, and red circles on the maps represent respectively the police agents, ambulance agents, and fire brigade agents. The bright green, dull green, and black circles represent healthy, hurt, and dead civilians in that order. The yellow, orange, and maroon colors represent the increasing intensity of fire in buildings; black represents completely burned and destroyed buildings. There are two special types of buildings: the

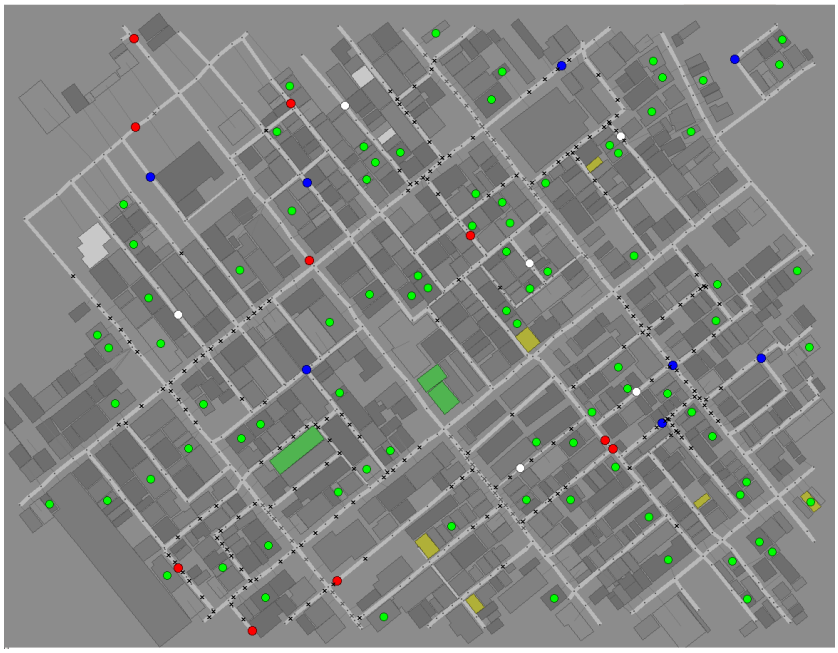

**Fig. 1.** The map of Kobe in the simulator

green buildings are refuges where saved civilians are taken, and the white buildings are centers. Cross marks on roads represent blockades. The simulation runs for 300 time steps.

The major contributions we make in this paper are: (1) novel algorithms for our team of agents, MinERS (Minnesota Emergency Response Squad). (2) an empirical study of performance of MinERS, which includes comparison with top agents from the 2009 RoboCup competition.

The rest of this paper is organized as follows. In the next section, we describe related work. This is followed by a description of the algorithms we use for each agent type, and experimental results. We conclude by outlining future work.

## 2 Related Work

In the search and rescue simulation there are many tasks, such as saving civilians, clearing the rubble, and extinguishing fires, but there is a limited number of agents, each with limited capabilities, so task allocation is the core problem.

In [3], three ways to solve the coordination problem are proposed: decentralized mutual adjustment, centralized direct supervision, and environment partitioning. The decentralized approach is more flexible but not always preferable. A hybrid approach between the decentralized and centralized approach is proposed in [4].

Evolutionary reinforcement learning is used by [5] at the ambulance center in order to decide how many ambulances should co-operate to save the civilians buried in a building. LA-DCOP, a low communication distributed constraint optimization algorithm, is used by [6]. Combinatorial auctions are used by [7]. Agents inform the police center about their rubble clearing needs. The police center passes the information to all the police agents, who submit bids. Combinatorial auctions, also used by [8], achieve optimal task allocation but require large computational power and message bandwidth. The paper compares auctions with a distributed mechanism using localized reasoning. According to the authors, the distributed approach achieves satisfactory results with low computation power and minimum messaging.

Both [3] and [7] suggested partitioning the disaster space among agents. In both cases the partitioning is pre-determined and is homogeneous. Such an arrangement can result in partitions that have a drastic difference in the number of roads in each partition. A more powerful partitioning strategy based on the degree of blockades on the roads is proposed in [3], but such approach requires massive real time survey of the environment by all the agents, which is too costly. We also use clustering, but we try to generate uniformly sized clusters.

In [9] coordination is addressed with coalition formation, solving the problem as a distributed optimization problem via DCOP and the Max-Sum algorithm, and including spatial and temporal constraints. However, the approach is not evaluated using RCR.

## 3 Our Approach to Decision Making and Coordination

A significant portion of the decision making and coordination among our agents is done through the centers. Specifically, the centers provide (1) communications among agents, (2) clustering algorithms, (3) support for auctions for task allocation to police and ambulance agents, (4) computation of probability distributions of presence of buried civilians over the buildings in the city.

### 3.1 Communications

The simulator limits severely the communications. Several communication channels are provided for the agents to send messages. A message has a maximum length of 256 bytes, so compression is needed to fit in more information into the message. A field agent can receive at most 4 messages per cycle and send a maximum of 4 messages, determined by the channels to which the agent is subscribed. Since messages are broadcast, however, this means that multiple agents sending on a channel will quickly overflow the agents capacity to receive messages. To avoid this we use round-robin to limit the cycles in which agents can send messages.

In addition to buffering messages sent by field agents, we also use message senescence so that the messages that get through tend to be the most important ones. At the start of the simulation, agents have to share a lot of information they sensed, but as the simulation proceeds the amount slowly drops. The buffering helps share information even when the channels available are not sufficient. Messages about cleared roads, fires, and civilians found are passed from the centers to the field agents to keep their local knowledge up to date.

### 3.2 Clustering Algorithms

The centers partition the city into clusters of roads and buildings based on the Manhattan distance between roads and between buildings, using the $k$-means algorithm, as implemented in CLUTO (CLUStering Toolkit). Figure 2 shows an example of partitioning the city for the police agents. For police agents, the number of clusters is half the number of police agents.

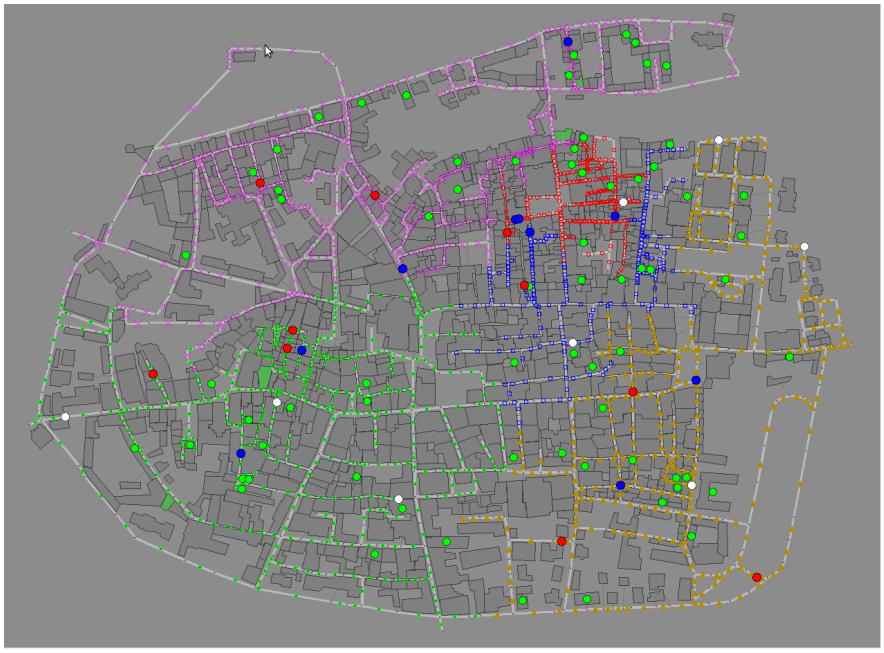

**Fig. 2.** Foligno map with roads divided into clusters for the police agents

The Ambulance Center partitions the city into areas of the same size and assigns each to an ambulance. The number of clusters is determined by the size of the city. Ambulances switch to a different unexplored area as soon as they finish exploring their own area, so spreading out over the city.

Fires start at multiple locations and spread to nearby buildings. This results in the formation of clusters of buildings on fire. The Fire Station partitions known fires into clusters of buildings, with a number of clusters equal to half the number of available fire brigades. This enables sending the fire brigades where they are most needed.

### 3.3 Auctions

Unlike fire agents where every unattended fire must be put out as promptly as possible, police and ambulance agents need to determine which portions of the city to attend to first to minimize civilians' deaths. They do this using auctions.

We use Sequential Single Item Auctions [10], which have the advantage of providing a better solution than the basic parallel auction [11] that assigns tasks to agents directly. In Sequential Single Item Auctions, a bid for a task accounts for the cost of previous tasks the agent has on its plate. This ensures that no agent is overloaded with all the tasks and ensures that tasks get addressed in a timely fashion.

The major challenge in implementing the auctions is due to the communication restrictions. Since each agent can only send and receive four messages per timestep, auctioning directly to agents would require multiple timesteps during which agents could instead be productive. We work around this by setting up a proxy agent that handles the actual auction process. The proxy is created by the center and it tracks the current and expected positions of the agents at each timestep. It also tracks whether agents are idle or busy at the specific timestep. Using the proxy introduces some error in the bidding process, since an agent may not be at the exact predicted location, but the error is bounded by the time taken by the agent to get back to that location, and is therefore never more than 2 timesteps.

The auction is implemented as follows:

1. Field agents send requests to the center for different tasks as they sense them.
2. The centers sort the requests with the most critical request first.
3. The centers activate the agent proxy with the requests to complete the auction.
4. The agent proxy assigns tasks created from the requests to agents that are not presently at work. Task assignment is done by estimating how long it would take for the agent to travel to that location, and then to perform the task. For ambulances the costs includes the cost of carrying civilians to a refuge.
5. After a task assignment, the proxy saves the cost and uses it when computing the total cost for the next task.
6. When all tasks are assigned, the auction stops and the centers send out the list of assignments to the agents.
7. The proxy continues to monitor agents after assignment, and marks them as idle when their tasks are completed.

### 3.4 Probability distributions

The Ambulance Center maintains a probability distribution of civilians over all buildings. This distribution is populated using sense messages received from field agents. The ambulance teams are periodically sent this information so that their local knowledge also gets updated.

We model the process of determining the probability of occupancy of a building as a Markov Process, described by the state and current inputs. Using this, we arrive at the following probability update rule for civilians heard (civilians seen are identified precisely; heard information only provides a range within which the person is probably present, and no other data). Given:
$a$ = a person is in a building nearby
$b$ = a person can be heard, so $P(\neg b|\neg a) = 1$, and it is specified that $P(\neg b|a) = 0.9$
$c$ = a person is in this particular building, so $P(a|c) = 1$.
$P(c)$ is the probability of occupancy for each individual building, which we desire to

compute.
$P(a) = 1 - \prod_{i=1}^{n}(1 - P_i)$, where $P_i$ is the probability of building $i$ having any occupants and $n$ is the number of buildings in hearing range. Then we have the following:

$$\begin{aligned} P(a|\neg b) &= P(\neg b|a) \times P(a)/(P(\neg b|a) \times P(a) + P(\neg b|\neg a) \times P(\neg a)) \\ &= 0.9 * P(a)/(0.9 * P(a) + (1 - P(a))) = 0.9 \times P(a)/(1 - 0.1 \times P(a)) \\ P(c|\neg b) &= P(c|a) \times P(a|\neg b) = 0.9 \times P(c)/(1 - 0.1 \times P(a)) \\ P(c|b) &= P(c|a) \times P(a|b) = P(a|c) \times P(c)/P(a) = P(c)/P(a) \end{aligned}$$

Because of communication limitations, messages may be received by the center out of order. Each time a message arrives out of synchronization, all messages from that point on are reexamined and the probabilities recalculated. By doing this we can treat the sense messages received as dependent only on the state at that time step, given the knowledge available at that time step. We thus satisfy the Markov property by maintaining independent behavior from other time steps, and hence satisfy the requirements for the Bayesian network update rule.

The probability of hearing a civilian per cycle is only 0.1 (10%). However, because there are multiple field agents and their information is pooled by the center, the possible location of any civilian heard can be narrowed down to a few buildings very quickly. Empirically over the course of 300 timesteps, all the living civilians get located.

## 4 Field Agents Design

Since field agents have limited communications they cannot coordinate efficiently. We rely on the centers to provide coordination functionality.

When the simulation starts, the agents get the map of the city but debris and the resulting blockades are visible only when an agent is close to them, i.e. none of the agents has a comprehensive view of the complete disaster situation. At the field agent level, agents rely on low-level coordination strategies to determine where they need to go. They use their knowledge of the starting map and assign themselves to specific regions to explore, ensuring that there is limited duplication of effort between them. They rely on the centers to provide them with more detailed information on pending tasks. In what follows, we describe the basic design of the field agents.

### 4.1 Police Agents

Police agents are responsible for clearing blockages on the roads which hamper the movement of ambulance and fire brigade agents. The cost to clear the debris from a road segment is directly proportional to its amount. Debris removed from the road can reappear as the simulation proceeds. While the debris clearance does not directly affect the score in the competition, it affects how quickly fire brigades and ambulances can reach a given area. Thus we need police agents to respond quickly to the needs of the other agents and at the same time to ensure that the city is cleared of blockades.

The city is partitioned into clusters, as shown earlier in Figure 2. Once the partitions are generated, each agent assigns itself, based on its unique id, in a round-robin fashion to a partition so that each partition has at least one agent. Each agent needs to visit all

the roads in its cluster to clear them. We solve this as a Travelling Salesman Problem (TSP). We first transform the road network into a graph where each vertex corresponds to a street segment and each edge to an intersection. We then convert the graph into a complete graph by using the all pairs shortest path Floyd-Warshall algorithm [12]. which generates a $n \times n$ distance matrix, consisting of the shortest paths between all $n$ vertices. We use the distance matrix as a complete graph for the Approx-TSP-Tour algorithm [12], which generates a tour whose cost is no more than twice the minimum spanning tree's weight.

In addition to following this tour, the agents respond to help requests from fire brigades and ambulances as and when they arrive. These requests are assigned by the Police Office using auctions. While responding to requests, the police agents save their previous path and return to the portion of the tour they were on after the requested blockade is cleared. This ensures that the agents complete their tour and attend to emergency requests in a timely manner.

### 4.2 Ambulance Agents

Ambulance agents are responsible for saving civilians that are injured or trapped. Civilians' conditions deteriorate over time, hence they have to be located and rescued as soon as possible.

Ambulance agents have two types of tasks:

1. Gather information. This involves counting the number of civilians trapped, their approximate location, and judging how soon and how much assistance they need.
2. Rescue civilians. Once a civilian is located, the ambulance has to remove the rubble trapping him, load him and take him to a refuge.

Civilians can occasionally be heard when within a hearing range, but locating them takes time because each building in their proximity has to be visited. It is often physically impossible to visit all the buildings in the timeframe given.

The location of a civilian may be known but inaccessible due to a blockade or fires in the area. Ambulances have to weigh waiting for the blockades to be cleared and fires to be put out against saving other civilians who may or may not be in as critical a condition. Coordination with other ambulances can help in speeding up the process.

Since the information available to each ambulance is insufficient to determine which civilians to rescue first, ambulances bid on tasks with the Ambulance Center, as described earlier, and also maintain a local copy of the ambulance center's probability map. Police and fire agents help by exploring buildings when their own tasks are completed. This speeds up the exploration process and allows the ambulances to concentrate on rescues instead.

### 4.3 Fire Brigade Agents

Fire brigades and the fire station are responsible for getting fires under control. The spread of fire depends upon wind speed and wind direction. Due to fire, buildings get damaged and collapse resulting in blocked roads. Additionally, fire can result in civilian death and agents getting hurt.

A critical responsibility of the fire agents is to maintain an accurate representation of the fire locations and intensities across the city. All field agents of each type report to the Fire Station the id and fire intensity of any building they sense is on fire, as well as the time they discovered it. The Fire Station keeps a list of all known buildings on fire, and updates each building's fire intensity as new data are collected. It uses this information to generate brigade assignments.

Since fires start in separate locations and spread outwards, the buildings on fire tend to form clusters around their respective points of origin. Since smaller clusters of less intense fires are much easier and vital to extinguish, we weigh each cluster as the sum of the squared intensities of the fires for each building in the cluster, and then invert the weights by subtracting them from the highest-weighted cluster. To ensure that each cluster gets addressed, one brigade is allocated per cluster, and the rest are assigned in proportion to the inverted weight of each cluster compared to the total weight of all the clusters. Within each cluster, brigades are assigned to first address the less intense fires using a similar weight inversion method using the square of the fire intensity.

Communication about fire locations and intensity of fires is often delayed, so the extinguish assignments may be based on slightly outdated information. Since fires grow in intensity over time, the delay often causes agents to be directed to extinguish a building where the fire is too intense. To mitigate this problem, when the brigades reach their target building, they search for nearby fires that would be extinguished more easily. This allows the agents to handle their assigned fire cluster without wasting time and water on hopelessly intense fires.

The decision-making process of the fire brigades at each timestep is as follows. The first thing a fire brigade does is look for civilians within its sensing radius. It then sends a message to ambulances indicating whether or not there are civilians in need of saving in the area. The second step is to check if the agent is currently inside a refuge. If so, it checks if its water level is less than full and then stays inside the refuge for this timestep.

The next thing the agent checks is if it is currently located on a road. If so, it checks if it has encountered a blockade, impeding its progress. If the agent has been blocked, and it has a current goal that it is working towards, the D* Lite replanning method is invoked to recompute a new path. If the agent does not currently have a goal, it simply moves into a nearby building for this timestep so it can get out of the way for police agents that may be arriving to clear the blockade.

The fourth check is whether or not the agent is currently sitting inside a building, and furthermore if that building has caught fire. If this is true, the fire brigade immediately attempts to escape from the dangerous situation. Next is a check of the agent's current health. If it detects that it is in danger of dying, it immediately moves to the nearest refuge to heal. Similarly, if it detects that its water level has reached 0, it goes to the nearest refuge in order to replenish its tank.

The last check involves looping through any commands it has received from the fire station to extinguish buildings that have caught fire. If it is close enough to the building requested, it checks the local area to determine the most easily extinguishable fire in reach, which it then extinguishes. Otherwise it plans a path to the building and moves on that path. Once the station commands have been exhausted, it loops through an internally maintained list of buildings it has seen that were on fire in a similar way.

If the decision making process falls through all of these checks, it means that the fire brigade has nothing to do. If the simulation is still early, i.e. less than 1/3 of the way through, this may just mean that the fire brigade has not explored enough. In this case, it moves randomly. However, towards the end of the simulation, this most likely means that all fires have been extinguished. If so, the fire brigade responds to requests sent by ambulances to explore unknown buildings to help discover trapped civilians.

Initially, we used A* search to plan paths for our fire brigades. Although optimal, this method failed to account for blockades and other path obstructions, and agents used a substantial amount of their processing time computing and recomputing new paths. To get better results, we have chosen to use the D*-Lite algorithm [13]. Its main focus is goal-directed navigation in unknown terrain. D*Lite performs better than A* because it computes optimal paths (the same paths that would be computed using A*) more quickly and more efficiently by modifying previous search results locally. With A*, sometimes search would not complete in the brief amount of time that an agent had to make decisions, resulting in agents not moving at all for several cycles at a time.

## 5 Experimental Results

Police agents contribute to the competition score only indirectly by clearing roads for ambulances and fire brigades. Therefore, their ability to clear blockages is the best measure of their performance.

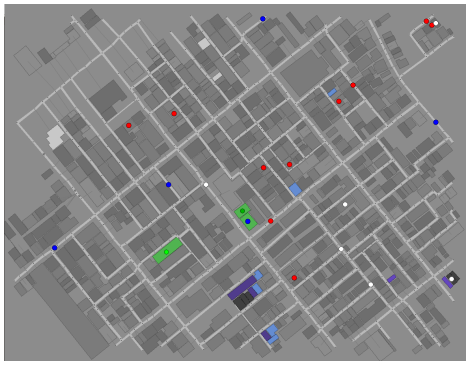

**Fig. 3.** MinERS at the end of simulation on Kobe. Very few buildings are burnt and no civilians are dead

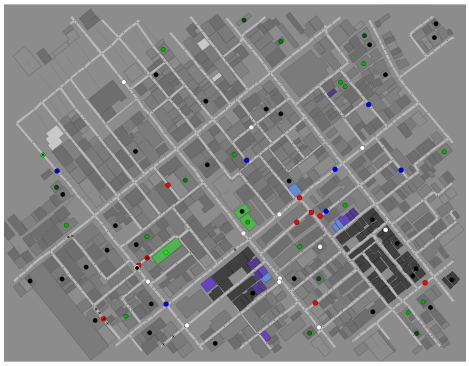

**Fig. 4.** Sample agents at the end of simulation on Kobe. More civilians are dead and buildings burnt

The simulator provides baseline agents, which we refer to as “Sample”. These agents perform a random walk over the map, and handle all problems encountered while on the random walk. Figure 3 and Figure 4 show the status of the Kobe map towards the end of the simulation with MinERS and Sample agents respectively. It is clear from the figures that our fire brigade agents are able to control the spread of fire better.

Comparisons of our police agents with the Sample police agents are in Table 1. In these experiments we use only police agents and blockades. This was done in order to evaluate the performance of the police agents without interference from other agents. Typically, Sample police agents work clear blockades till the end of simulation, while the MinERS Police Agents clear the blockades much earlier.

**Table 1.** Number of blockages cleared by Sample and by MinERS Police Agents

| Map | Sample | MinERS |
|---|---|---|
| Kobe | 227 | 244 |
| Random Small | 536 | 600 |
| Foligno | 264 | 377 |
| Virtual City | 247 | 270 |

We compared the performance of MinERS vs. agents which have competed in the 2009 competition and vs. the Sample agents. We chose three other teams for comparison, two of which (MRL and Poseidon) have participated regularly in the competition and were finalists in 2009, and one (Lotus) was a new team like ours. The scores of the four teams that were in finals were very close. MRL ranked second in 2009, and Poseidon third. We could not get the code of the competition winner to run.

Table 2 shows data on the maps used, including starting scores, number of agents, and number of civilians to be rescued. The maps drive the simulation – they contain all the information about the location of agents and civilians, ignition points (the points where fire will start) and blockades, and have different challenges for the different maps.

**Table 2.** The different maps used in the experiments and their properties.

| Map | Number of | | | | | | Initial |
|---|---|---|---|---|---|---|---|
| | Ambulances | Fire Brigades | Police | Buildings | Civilians | Fires | Score |
| Kobe | 15 | 10 | 8 | 740 | 144 | 6 | 178 |
| Random Small | 7 | 10 | 11 | 1219 | 66 | 6 | 103 |
| Foligno | 8 | 10 | 10 | 1085 | 70 | 4 | 99 |
| Virtual City | 7 | 6 | 11 | 1271 | 80 | 5 | 105 |

We show the mean and standard deviation of the scores of all the field agents on four maps in Table 3, obtained from 30 independent trials of each agent in each map. In the table we include two version of MineERS, one without using auctions and one using auctions. We note that the performance of MinERS is markedly better than that of the Sample agents on all the maps. The ambulance agents also succeed in transporting

**Table 3.** Comparison of performance of MinERS with other agents on maps with blockades

| Map | Sample | Lotus | MinERS no auctions | MinERS | Poseidon | MRL |
|---|---|---|---|---|---|---|
| | $\mu$ ($\sigma$) | $\mu$ ($\sigma$) | $\mu$ ($\sigma$) | $\mu$ ($\sigma$) | $\mu$ ($\sigma$) | $\mu$ ($\sigma$) |
| Kobe | 90.44 (8.84) | 115.86 (3.96) | 120.08 (9.97) | 138.18 (3.54) | 154.70 (3.06) | 159.21 (4.50) |
| Random S. | 55.58 (5.80) | 65.70 (2.60) | 71.55 (4.40) | 80.96 (2.29) | 89.70 (1.65) | 77.67 (9.87) |
| Foligno | 13.33 (1.61) | 15.58 (0.82) | 14.42 (0.92) | 18.18 (0.68) | 21.85 (0.83) | 22.91 (1.37) |
| Virtual C. | 32.72 (1.84) | 41.06 (1.59) | 37.31 (2.34) | 48.90 (2.28) | 60.14 (1.38) | 63.49 (2.05) |

a significantly higher proportion of the civilians to the refuge. The better performance is evident from the score difference between the maps.

Our agents perform better than the Sample agents and the Lotus team agents, however we need to make improvements to be competitive against the top agents.According to [9], some teams use specialized domain information, while we have kept our solutions generalizable.

We assume the scores for an agent on each map are normally distributed. We performed a one-tailed Welch's t-test (with $n = 30, \alpha = 0.001$) on our agents. We found that the difference in performance is significant at the 0.001 level ($p$ was on the order of $10^{-11}$ to $10^{-27}$ for the four maps we studied) for MinERS using auctions compared to MinERS without using auctions.

Auctions reduce the time taken to address each task – previously multiple tasks were getting neglected while agents converged on a single task. The auction and clustering mechanism enables the centers to ensure that tasks get proportional attention and no task gets neglected. This has been especially helpful since now police agents enable fire brigades to reach fires faster when they are easier to contain, and ambulances are able to explore more of the city faster.

The second major contributor in the improved score is the searching done by the fire brigades and police agents after their tasks are completed. This speeds up the rate at which the city is explored, and resulted in a larger number of civilians being rescued. A few points to note are:

1. Thanks to communication and coordination, our fire brigade agents are able to reach all the sites where fire breaks out. On the other hand, lack of communication results in idling of Sample agents despite task availability.
2. Our police agents locate and clear all the blockades in the Kobe city map, and consistently clear more blockades than the Sample agents.
3. Our police agents respond to requests to clear blockades from other agents, allowing them to start performing their tasks early on in the simulation.
4. Both the police and fire brigade agents respond to ambulance center requests to check buildings for civilians when they have completed their tasks. This relieves the ambulance agents of extra work and accelerates discovering and locating trapped civilians.

5. The ambulance agents are able to explore all the buildings in the Kobe city map, and locate and rescue all the civilians in that map by the end of the simulation.

## 6 Conclusions and Future Work

We have presented solutions to task prioritization and allocation among distributed heterogeneous agents in large scale urban search and rescue. We used the RCR simulator to develop and test our strategies for the rescue agents, and presented the results we obtained. Agents using our algorithms perform markedly better as compared to agents which work stand-alone. We have shown how the coordination algorithms we use at the center level have increased the rate at which our agents can successfully respond.

For the future, we plan on porting our algorithms to the new simulator that has been released this year and on adding further heuristics at the individual agent level. We will also be working on a closer integration of the different agent types, and to increase the interaction among the three centers.

## References


1. Kitano, H., Tadokoro, S.: RoboCup rescue: A grand challenge for multiagent and intelligent systems. AI Magazine **22**(1) (2001) 39–52
2. Takeshi, M.: How to develop a RoboCupRescue agent (2000)
3. Paquet, S., Bernier, N., Chaib-draa, B.: Comparison of different coordination strategies for the RoboCup rescue simulation. In: Innovations in Applied Artificial Intelligence. Volume 3029. Springer-Verlag (2004) 987–996
4. Mohammadi, Y.B., Tazari, A., Mehrandezh, M.: A new hybrid task sharing method for cooperative multi agent systems. In: Canadian Conf. on Electrical and Computer Engineering. (May 2005) 2045–2048
5. Martínez, I.C., Ojeda, D., Zamora, E.A.: Ambulance decision support using evolutionary reinforcement learning in RoboCup rescue simulation league. In: RoboCup 2006: Robot Soccer World Cup X. Springer-Verlag (2007) 556–563
6. Scerri, P., Farinelli, A., Okamoto, S., Tambe, M.: Allocating tasks in extreme teams. In: Proc. Int'l Conference on Autonomous Agents and Multi-Agent Systems, ACM (2005) 727–734
7. Bredenfeld, A., et al.: RoboCup 2005, LNAI 4020. Springer (2006)
8. Nair, R., Ito, T., Tambe, M., Marsella, S.: Task allocation in the RoboCup rescue simulation domain: A short note. In: Int'l Symposium on RoboCup. (2001)
9. Ramchurn, S., Farinelli, A., Macarthur, K., Polukarov, M., Jennings, N.: Decentralised coordination in RoboCup rescue. The Computer Journal (January 2009)
10. Koenig, S., Tovey, C., Lagoudakis, M., Markakis, V., Kempe, D., Keskinocak, P., Kleywegt, A., Meyerson, A., Jain, S.: The power of sequential single-item auctions for agent coordination. In: Proc. Nat'l Conference on Artificial Intelligence. (2006) 1625–1629
11. Dias, M.B., Zlot, R.M., Kalra, N., Stentz, A.: Market-based multirobot coordination: A survey and analysis. Proceedings of the IEEE **94**(7) (July 2006) 1257–1270
12. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms. Prentice Hall of India (2001)
13. Koenig, S., Likhachev, M.: D* lite. In: Proc. Nat'l Conference on Artificial Intelligence. (2002) 476–483