MinERS: Preliminary Report

Maitreyi Nanjananth, Abdul Rub Aamer Mohammed, Ankur Satyendrakumar Sharma, Srivenkatesh Kumar Vaithianathan, Alexander J Erlandson, and Maria Gini

University of Minnesota, Minneapolis, MN 55455
{nanjan,mohammed,ankur,svaith,erland,gini}@cs.umn.edu

Abstract. We describe the architecture and strategies used in MinERS (Minnesota Emergency Response Squad) and present preliminary results.

1 Introduction

The RoboCup Rescue Agent Simulation competition provides a platform for designing and testing disaster management and mitigation strategies where heterogeneous agents (Police, Fire Brigades, and Ambulances) co-ordinate with each other to deal with a simulated disaster scenario [1–3]. The overall objective of the competition is to save as many civilians as possible and minimize the damage to the buildings that are on fire, while at the same time clearing the rubble on the road for smooth routing.

We ran our experiments using the RoboCup Rescue release version 0.49plus, which is the version that will be used for the competition in 2009. We used a distributed setting where each type of agent ran on a different machine, while the kernel and all the simulators ran on the same machine.

2 System Architecture

The overall functionining of both ambulance and police agents is modeled using the Subsumption Architecture [4]. We now describe each type of agent.

2.1 Police Agents

Police agents are responsible for clearing rubbles present on the road. The presence of rubble drastically hampers the movement of ambulance agents and fire brigade agents over the map. Therefore, even though police agents do not directly participate in the process of rescuing civilians and dousing fire, they are inherently responsible for smooth execution of these tasks. There is also an implicit requirement that the rubble gets cleared off from the environment as quickly as possible since any delay will only slow down the routing of the other agents.

During the start of the simulation process all the police agents get the GIS information about the city. The GIS includes information about the roads in the city, their connectivity, length and the number of lanes they have. Rubbles, however, are randomly scattered over the city and the information made available is only on vicinity, i.e. none of the agents have a holistic view of the complete disaster situation. The cost required to clear rubble from a road segment is directly proportional to the degree of its presence on the road. Rubble once removed from the road can reappear as the simulation proceeds.

The task of clearing rubble can be distributed among police agents. Thus, any policy governing the behavior of the agents needs to ensure that police agents are optimally used, i.e. in the entire simulation life cycle they should be clearing rubble for a significant percentage of time.

[5–7] have proposed broadly two kinds of approaches to solve the problem. The first is to use a centralized approach, where the police center, also called police office, decides which police agent should clear which road. In this approach the police center gets information about rubble from police agents as well as other agents and then, based on some utility maximization function, each police agent is assigned a road to be cleared. Since the police center has a better visibility about the disaster scenario, it can make much more informed decisions. However, as the overall design is similar to a hub-and-spoke architecture, it suffers from the limitation that if the hub goes down for some reason, the entire architecture is rendered useless. The hub in this case is the police center that could be unreachable due to communication breakdown caused by disaster or the police center itself could suffer from a drastic damage due to disaster. Thus, there is a requirement for an alternative mechanism to limit the loss.

The second approach is a decentralized strategy where the police center is minimally or not at all used and each agent individually decides which roads to clear. The obvious drawback of this strategy is that none of the agents has a universal view of the disaster scenario, thus any decision made by an agent is mostly based on locally available information. However, since RoboCup Rescue simulates a disaster scenario where it is quite possible that there will be no communication with the police center, it is our conjecture that a decentralized strategy should outperform a centralized strategy. [6] claims that a decentralized strategy slightly outperforms the centralized one. For these reason, our initial implementation of the police agents follows a decentralized strategy.

Both [6] and [5] suggested partitioning the disaster space among agents. In both cases the partitioning is pre-determined and is homogenous. Such an arrangement can result in partitions that have a drastic difference in the number of roads in each partition. [6] mentioned a more powerful strategy for partitioning the space based on the degree of blockades on the roads. However, such an approach requires a lot of real time survey of the disaster environment by all the agents, which could be a costly task by itself.

We propose a more moderate strategy for partitioning the disaster space that is based on the spatial nature of the map. We generate partitions as clusters of roads based on the Euclidean/Manhattan distance between roads. We have implemented clustering using the K-Means algorithm. One of key criteria for this selection was its fast convergence to approximate clusters. As one of the input parameter for K-Means is the number of cluster to be generated, the number of partitions needs to be pre-decided. We select the number of clusters N_c to be $\lfloor N_p/2 \rfloor$, where N_p is the number of police agents.

This strategy guarantees that there is at least one agent per sector. We have used CLUTO (CLUStering Toolkit), a software package developed at the University of Minnesota, for clustering and analyzing the clusters.

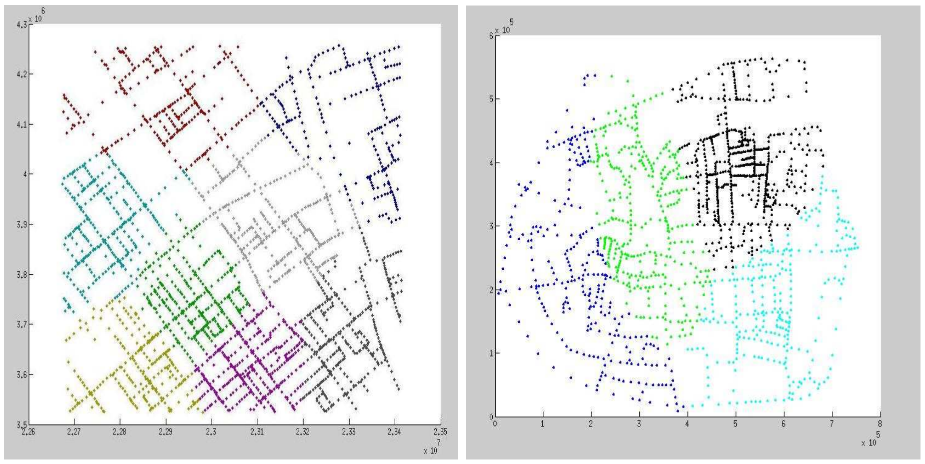

Fig. 1. (Left)Kobe 4 Map with roads divided into 8 clusters. (Right)Foligno Map with roads divided into 4 clusters. Roads are marked with different color and roads with same color belong to the same cluster.

Before the actual simulation begins each police agent has to (1) generate clusters of the roads and (2) generate the partition ID for itself. Each agent separately computes the clusters that will be treated as partitions in the real environment. In order to make sure that clusters generated by one agent do not differ from the clusters generated by any other police agent, we make sure that the structure of the input data to the clustering algorithm is the same as well as the seeds used by the random number generator within the clustering algorithm. Once the partition information is generated, the agent uses the following algorithm to figure out the partition it belongs to.

Algorithm to Generate Partition ID:

1. Iterate over all the agents and allocate agent to a partition if the agent is at a location in the partition and there is no other agent already allocated to that partition.
2. Iterate over all the partitions that did not get any agent assigned to them in the previous step and assign an un-assigned agent to each partition.
3. Use Round robin to assign the remaining un-assigned agents to the partitions.

Since this algorithm is invoked before the actual simulation has begun, each agent knows the exact location of the other agents and the algorithm will give a consistent result.

At each time step, every police agent does the following:

1. The agent may or may not be present in the assigned partition. In any case the agent tries to first move to the refuge in the partition. If there is no refuge in the partition, the agent tries to move to a randomly selected road in the partition.
2. Each agent then generates a "walk" over all the roads that are present in the partition and stores in memory. The "walk" confirms that agent has visited all the roads in the partition.
3. If there is a blockade at the location where agent is, the agent tries to clear the block. The blockade cleared by the agent can also be a priority task blockade.
4. If the agent has received a request to clear a blockade from another agent in the same partition, then the agent considers this as a priority task.
5. If the agent has a priority task then the agent tries to move to the location where the requesting agent has informed about the presence of the blockade. Otherwise it tries to complete the "walk" on the partition.
6. If the agent has completed its walk on the current partition, the agent selects a new partition.

Performance Evaluation of Police Agents The Robocup Rescue Agent simulation competition in 2009 uses a new vector based scoring mechanism. Unlike the previous scoring mechanism that took most of the decision based on the health point of the humanoids and did not put any emphasis on the degree of planning and co-ordination done by the agents, the current scoring considers factors like moving average utilization of fire brigade agents and ambulance agents, fire extinguished ratio, building area that has remained intact and other. However, there is no explicit metric measuring the performance of the police agent. We propose a mechanism for measuring the performance of the police agent. In the disaster situation the performance of a police agent comprises of the following parameters:

1. Cost of non-priority blockades removed from the road. This is a parameter that a police agent would like to maximize.
2. Total distance travelled by the police agent. This is a parameter that an agent would like to minimize. Even though in the RoboCup Rescue simulation environment there is no cap on the fuel consumption by the police cars, in a realistic sense there should be an upper bound to it, so a police agent should prefer some degree of localization while selecting target roads to be cleared.
3. Cost of priority blockades removed from the road. This implicitly addresses the requirement of co-ordination between different types of agents.

Let Cnp_i be the cost of clearing non-priority blockades by police agent i, Cp_i be the cost of clearing priority blockades by police agent i, and D_i be the total distance travelled during the life cycle of the simulation by police agent i.

The performance of a police agent i will be given by:

$$\eta_i = (Cnp_i + 1) * (Cp_i + 1)/(D_i + 1)$$

The overall performance, Pt, of police agents will be a summation of the performance values of all the police agents. So while comparing the overall performance of police agents on two simulation runs that are based on the same simulation environment i.e. same map and same initial conditions with respect to disaster, the one with higher Pt will be treated as better.

Table 1. Comparison of scores of MinERS agents with Sample agents

Map	MinERS	SampleAgent
Kobe (8-Runs)	29.61 (± 7.96)	2.36 (± 0.06)
Foligno	36.68	1.59
VC	353.09	8.831

The table 1 shows a comparison between the performance of police agents of Sample code and the police agents of MinERS for various maps. 8 Runs were made on the map of Kobe, while one run each was made for the other mentioned maps. The Pt of sample code is quite low as compared to the Pt of MinERS. This is because police agents of sample code do not deal with priority tasks.

2.2 Ambulance Agents

In the simulation environment, ambulance agents are responsible for saving civilians that are buried under a building after the disaster. At the start of simulation none of the ambulance agents has any knowledge of civilians that may be trapped or injured within buildings in the city. The first challenge for them is to locate the civilians in the buildings. Once an agent finds a civilian, the agent needs to first remove the rubble from the buried civilian. The cost of execution, i.e. time taken to execute, of this task is directly proportional the level of buriedness of the civilian. Once all the rubble is removed the civilian is loaded on the ambulance and then taken to refuge. Ambulance agents cannot remove blockades from roads, so the route chosen from the disaster location to the refuge should avoid blockades. Otherwise, the ambulance agent has to raise a priority task to a police agent.

Unlike both police and fire personnel, who can identify an exact location when its ID is provided, the ambulance agent does not have this ability - civilians are usually completely unknown until they are found (either heard or seen). Thus the ambulance teams need to search every building before they can be sure of finding everyone (dead or alive). Since people alive may be hurt and losing strength over time, ambulance agents need to locate and retrieve them as soon as possible, followed by retrieving any dead bodies that may be found.

The default way to do this would be a random search. Unfortunately, this is time consuming and can miss trapped people because of distance issues. The simulation provides an additional handicap not seen in real life: when someone is shouting for help, in real life we can usually pinpoint the general direction of the person - in simulation we cannot. Thus, we seek to model graceful degradation in the ambulance agents, allowing them to perform their tasks effectively even in the face of deteriorating communications and infrastructure.

The ambulance strategy operates on three levels:

1. Greedy local level - Rescue the nearest civilian, and start a building-by-building search for humans in the vicinity. Use A* search to find the nearest human, with parameters tuned to optimize search and travel time (for example, ensuring roadways that were previously known to be blocked are not used when in a hurry)
2. Neighborhood approach - Tune the search to rely on data from other agents. This usually means asking fire brigades and police agents to provide information about civilians in their vicinity.
3. Central level - use the Ambulance coordination center to store and send out information about nearby humanoids, and use it to determine where to go, and what roads are clear.

Each ambulance maintains a probability distribution indicating the likelihood of a building being occupied by a civilian in need of help. Information received while exploring and from other agent messages is used to update this probability distribution. The agents start with a uniform distribution over all the buildings, using an arbitrary prior probability of occupation.

The rescuing of civilians requires coordination - an agent that has found multiple civilians should be able to coordinate with other agents to determine the order in which to address the needs of the civilians to minimize further injury to them. The ambulance agents coordinate through a hierarchical structure: the agents select two leaders, and follow the leaders' direction in terms of which civilians to rescue. Leaders may be switched out if the agents encounter problems. If communication with a leader is no longer available (for example, if the leader is dead) then the agents assign a new leader amongst themselves. An agent following a leader will start working to save a nearby civilian if it becomes clear that the leader can manage the remainder of its task independently.

The process of loading and moving civilians needs roads to be clear until refuges can be reached. This requires a method of finding alternative routes to destinations, since delay of even a small amount of time can result in the death of a civilian who might have otherwise survived. This has been done by maintaining paths and ensuring that a blocked route is not chosen until it is known that the route is now cleared and navigable. In addition, other ambulance teams need to be up-to-date on the status of the ambulance agent and whether the civilians have been successfully rescued. The communication system is used to achieve this coordination and is geared to minimize message loss and maximize the information contained in each message sent.

The ambulance agents do not rely on the center as heavily as is done in [8] and [9]. At the center level, the ambulances use a system of auctions to distribute tasks. This works well at the top-level for coordination as it takes some of the computational burdem off the center, moving it to agents that already have to perform those computations to complete their tasks. We will use this in parallel to using the probability distribution to model the city and the need for ambulance agents in different regions.

2.3 Fire brigade Agents

During a high-intensity earthquake, there is a high chance of buildings being razed to the ground. There is also a chance of fire erupting in the buildings and spreading to nearby locations. In the robocup rescue simulation project, the firesimulator is responsible for randomly selecting buildings which should be ignited. Once a building catches fire, its neighboring buildings in the block have a high chance of catching fire too. Fires cause damage to the buildings which directly effects the score. Hence, the responsibility of the fire brigade agents is to douse the fire, as quickly as possible so that it does not spread further.

When the simulation starts, the fire brigade agents do not have any idea of where the fire is spreading. They depend upon combing the area to find the location of fire. In addition, agents of other types i.e., ambulance and police agents, can provide fire related information to the fire brigade agents. This information has to be processed so that appropriate number of fire brigade agents are assigned to each cluster of buildings on fire.

1. We noticed that fire occurs in clusters of buildings and blocks spreading from one building to its neighbor and one block to the one next to it. As in [10], we use k-means to determine appropriate clusters. Currently, it is the responsibility of the center agent to use k-means for clustering but this process will not work in case of center failures. The center agent also draws a convex hull around the cluster and distributes agents around the hull so as to contain the spread of fire. We believe that having agents surround a cluster is the best way to contain the spread of fire to nearby buildings.
2. The distribution of agents among the clusters and within the clusters is in itself an interesting area of study. Within a cluster, we use the output of the convex hull algorithm to distribute the available agents within the cluster. To distribute agents among clusters, we use the average fieriness of the buildings in a cluster to allocate the number of agents to attend to a cluster, as in [10].
3. Each firebrigade agent maintains two lists of buildings that it needs to attend to. One is a list of buildings which the center wants the agent to work on. If the center fails, this list would eventually be empty. The agent would switch to its local list of buildings that are on fire. Currently, the local list is a priority queue with buildings with highest fieriness attended to first. This is a simple strategy that we are currently following to gracefully degrade in case of center failures. We are still working on a strategy to elect a leader.

4. One issue which has still not been addressed in our implementation is that fire starts at places where there has been no combing operation and spreads rapidly to surrounding buildings. This problem is exacerbated as there is no way for agents to reach the location as the blockades have not yet been cleared.

Table 2. Comparison of scores of MinERS agents with Sample agents

Map	MinERS	SampleAgent
Kobe	67.273	57.819
Foligno	63.868	62.396
VC	38.999	35.969

2.4 Combined Experiments And Results

A comparison of results of MinERS agents vs. Sample agents is shown in Table 2. On the Kobe map, we also conducted 5 experiments each with both the sets of agents. The average score of MinERS was 60.797 and the average score of sample agent was 49.979. The results show our agents perform slightly better than the sample agent and we expect to improve them further as we continue developing them.

3 Related Work

The RoboCup Rescue simulation environment has been used not only for competition but also as a test-bed for research purposes. The core research problem presented by a simulated disaster environment is that of task allocation. In the disaster environment there are many tasks, like saving the civilians, clearing the rubble and dousing the fire, but there is a fairly small number of agents each with limited resources and capabilities to perform the task. The problem gets even more complicated because some of the constraints are temporal in nature, like the health point of the humans and the spread of fire in buildings.

Multiple approaches have been used to tackle the problems, ranging from machine learning (e.g, [9], to distributed constraint optimization (e.g, [11]), to combinatorial auctions (e.g., [5]).

[9] use an Evolutionary Reinforcement Learning at the ambulance center in order to decide how many ambulances should co-operate to save civilian buried under the rubble of a building. Thus they have treated this as a machine learning problem. [11] have treated this as a constraint optimization problem. They have proposed LA-DCOP, as low communication distributed constraint optimization

algorithm. [5] have treated the problem of clearing blockages by police agents as a combinatorial auction problem. They have used a single auction wherein other agents inform the police office about various tasks and the police office tells this to all the police agents. Police agents then submit bids on these rubble clearing tasks. The best agent for each task is assigned the task. Only one police agent is required to remove a blockade.

4 Conclusions and Future Work

We have described the high-level architecture and strategies we are using for agents in MinERS and reported preliminary results. Since this is the first time we enter the competition, we had to spend significant time coming up to speed.

References

1. Kitano, H., et al.: Robocup-rescue: Search and rescue for large scale disasters as a domain for multi-agent research. In: Proc. of IEEE Conference SMC. (1999)
2. RoboCupRescue Technical Committee: Robocup-rescue simulator manual version 0 revision 4 (2000)
3. Takeshi, M.: How to develop a robocuprescue agent (2000)
4. Brooks, R.A.: How to build complete creatures rather than isolated cognitive simulators. In: Architectures for Intelligence, Erlbaum (1991) 225–239
5. Bredenfeld, A., et al.: RoboCup 2005, LNAI 4020. Springer (2006)
6. Paquet, S., Bernier, N., Chaib-draa, B.: Comparison of different coordination strategies for the robocuprescue simulation. In: Proc. Int'l Conf. on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems. Volume LNAI 3029., Springer-Verlag (2004) 987–996
7. Nair, R., Ito, T., Tambe, M., Marsella, S.: Task allocation in the robocup rescue simulation domain: A short note. In: International Symposium on RoboCup (RoboCup'01). (2001)
8. Paquet, S., Bernier, N., Chaib-draa, B. In: Damas Rescue Description Paper. Springer-Verlag, Berlin, Heidelberg (2004)
9. Martínez, I.C., Ojeda, D., Zamora, E.A. In: Ambulance Decision Support Using Evolutionary Reinforcement Learning in Robocup Rescue Simulation League. Springer-Verlag, Berlin, Heidelberg (2007) 556–563
10. 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
11. 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, New York, NY, USA, ACM (2005) 727–734