Minnesota Emergency Response Squad (MinERS) Team Description Paper

Maitreyi Nanjananth, Sean Andrist, Alexander J Erlandson, Jonathan Hsiao, Aravind Ragipindi, and Maria Gini

University of Minnesota, Minneapolis, MN 55455
{nanjan}@cs.umn.edu

Abstract. We describe the decision processes of the MinERS agents. We developed different strategies for the different types of agents. Our police agents cover the city in minimal time to clear blockades. Our ambulance agents determine where to go using a Bayesian network that estimates civilian locations based on sensory information received. Our fire brigades use a weighted clustering algorithm to select fires that need attention and contain them. We examine how effective these strategies and algorithms are and compare their performance against the sample 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.

The major contributions we present here are:

1. algorithms for the different types of MinERS agents. We have created different algorithms for the police, ambulance, and fire brigade agents. We have tailored the algorithms to handle uncertainty and lack of information as well as failures in parts of the system.
2. an empirical study of performance. The study includes comparison of MinERS with the recently released agents from the 2009 competition and against the sample agents provided with the simulator.

We used the rescuecore package as our starting codebase, and based our agents off the sample agents provided in that package. We have made considerable modifications to the agents originally provided. We also present the changes we have made in comparison to our submission from last year, and the improvements we expect to make. We are at present modifying the system to work properly with the new Rescue Simulator version, hence the preliminary results we present are based on the simulator from the 2009 competition. The rest of this paper is organized as follows. First we present the algorithms used for each agent type, then we analyze the performance of the entire team, and outline changes we are making to our agents and future work.

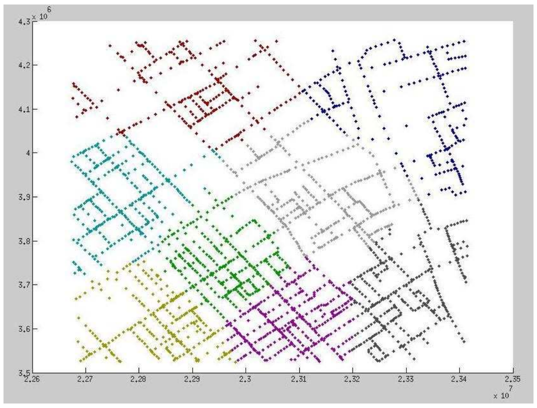

Fig. 1. Kobe Large Map with roads divided into 8 clusters. Roads of the same color belong to the same cluster.

2 Police Agents

We partition the city into clusters of roads and assign police agents to each cluster. To ensure roads are blockade free, each agent needs to visit all the roads in its cluster. This requires each police agent to:

1. Partition the city into clusters of roads. We use k-means [1] in CLUTO (CLUStering Toolkit) using the Euclidean/Manhattan distance between roads. Figure 1 shows an example of partitioning. The number of clusters N_c is $\lfloor N_p/2 \rfloor$, where N_p is the number of police agents. Each agent separately computes the clusters. To ensure that agents generate the same clusters, the structure of the input data and the seeds used by the clustering algorithm are the same.
2. Assign itself to a partition in a round-robin fashion so that each partition has at least one agent.
3. Solve the Travelling Salesman Problem (TSP) in its partition. To do this, we transform the road network graph to 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 [2–4], which generates a $n \times n$ distance matrix, consisting of the shortest paths between all n vertices. The distance matrix is used as a complete graph for the APPROX-TSP-Tour algorithm [4], which generates a tour whose cost is no more than twice the minimum spanning tree's weight. We compute the Floyd-Warshall algorithm for the entire map in the initialization phase. However, since the run time of the algorithm in some cases is too close to 300 seconds, we developed a shared memory parallel implementation [5]. The initial implementation was made using Java-Threads, but the final squeeze was made using the C-based posix threads, also known as p-Threads.

3 Ambulance Agents

Ambulance agents have to save civilians that are injured or trapped. Since the number of civilians and their locations are not known, the first challenge is to locate them and the next challenge is to rescue them as soon as possible to avoid deterioration of their health. Ambulance agents have two types of tasks:

1. Gather information. This involves counting how many civilians are trapped, approximately where they are, how trapped/hurt they are, and judging how soon and how much assistance they need.
2. Rescue civilians. This is the primary ambulance task, but it depends on the effectiveness of the exploration. Efficiency in rescue is critical, because trapped civilians may die if help does not arrive quickly.

Each ambulance maintains a probability distribution indicating the likelihood of a building being occupied by a civilian in need of help. We model the process of determining the probability of occupancy of each building as a Markov Process, fully 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 within hearing range
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 $P(a|c) = 1$.
Then, $P(c)$ is the probability of occupancy for each individual building, which we desire to compute, and $P(a) = 1 - \prod_{i=1}^{n}(\neg P_i)$, 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:

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

Unfortunately, the communication delay in the system means that new messages may arrive earlier than older messages, violating the requirement for Markov Processes that the history not be relevant to the state. We work around this by ignoring earlier messages about a civilian or building that has already received a later update. This allows the state description to be maintained without the need to maintain a history, at the cost of a small loss of information.

The center serves primarily as a message coordinator – messages about cleared roads and civilians found are passed on to ambulance agents to keep their local knowledge up to date. The center also tracks rescued civilians and tries to send their positions to the ambulances in a timely manner so that ambulances do not waste time moving to rescue an already rescued victim. This provides a reduction in the time spent exploring by the ambulance agents.

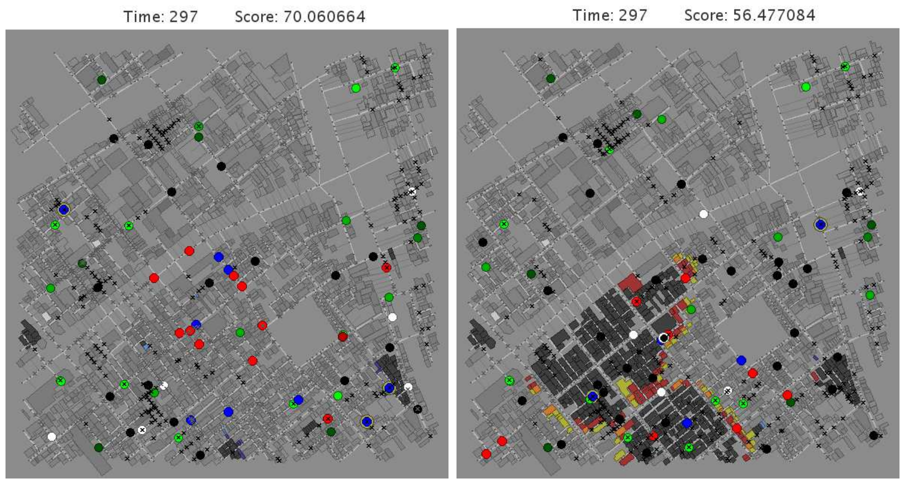

Fig. 2. MinERS fire agents (left) and sample agents (right) on Kobe Large, with 12 fire brigade agents and blockages enabled

4 Fire Brigade Agents

Fire brigade and fire station are responsible for getting the fire under control. Fires start at multiple locations and spread to nearby buildings. This results in the formation of a cluster of buildings on fire. The spread of fire depends upon wind speed and wind direction. Fire has a direct impact on the score of the simulation both directly due to building damage and indirectly due to civilian deaths – the greater the area of the damaged buildings, the greater the number of dead civilians, and the lesser the score of the simulation.

The main issues fire brigades agents face are:

1. The agents are unaware of the locations of fire and have to discover them.
2. Response time is of utmost importance as it limits the spreading of fire. The agents are most effective when they reach the fire in early stages.
3. A fire site may not be accessible due to falling debris.
4. The refuges where the fire brigades refill water may not be accessible due to road blockades.
5. Fire starts at multiple locations, so task allocation is critical.

We determine the cluster of buildings on fire using the k-means algorithm using the location of buildings and the Manhattan distance between them as parameters. We allocate fire brigade agents to each cluster depending upon the average intensity of fire of buildings in the cluster. To avoid the spread of fire to more buildings, we draw a convex hull [6] around the cluster and distribute the agents around the hull to contain the spread of fire. Figure 2 shows fire brigade agents extinguishing fires 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.

5 Results on Combined Experiments

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. We compared the performance of MinERS police agents with the sample police agents, by showing the number of blockages cleared in different maps:

Map	Sample Agents	MinERS Agents
Foligno	245	364
Virtual City	1247	1676
Kobe	1369	1578

In the police agent performance experiments we used only police agents and blockages. This was done in order to evaluate their performance without interference from other agents. Typically, the sample police agents work till the end of the simulation to clear blockades, while the MinERS police agents clear the blockades much earlier.

We compared the results of all the MinERS agents vs. agents which have competed in the 2009 competition and with the sample agents provided in the simulator. We chose three other teams for comparison, two of which have participated regularly in the competition and were finalists in 2009, and one new team like ours. We performed comparably better than the other new team, but need a lot of improvements to compete against the two finalists.

We show comparison experiments on four of the maps provided with the RCRS package. The maps drive the simulation – they contain all the information about the location of agents and civilians, points where fire will start, and blockades, and have different challenges:

	Number of						Initial
Map	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

Here are the comparisons on each map, both with and without blockades:

	Without blockades					With blockades				
Maps	MinERS	Sample	Lotus	MRL	Poseidon	MinERS	Sample	Lotus	MRL	Poseidon
Kobe	147.9	117.8	127.1	161.5	158.9	127.7	98.0	125.9	157.4	156.0
Random Small	26.7	21.9	24.1	30.1	33.6	16.9	15.3	18.1	26.0	24.9
Foligno	93.8	72.9	80.4	84.7	98.2	83.3	63.0	74.9	89.3	93.1
Virtual City	62.8	44.4	54.3	78.2	76.1	45.3	40.5	51.9	70.7	71.3

Disabling blockades gives free access to the disaster sites, and so we can measure the performance of the ambulance and fire brigade agents without any dependency on the police agents. We clearly see that the agents perform better when there are no blockades. 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 saving a significantly higher proportion of civilians.

The results show our agents perform better than the Sample Agents and we expect to improve them further as we continue developing them. The results show also our team performing better than the Lotus team on average when there are no blockades. The coordination of our team is still not fully developed, specifically with the police agents, hence when the agents need to coordinate (with blockades) our performance dropped by more than the other agents' performance. 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.
2. Our agents are able to clear the entire Kobe map in 175 cycles when blockades are disabled as compared to 270 cycles when blockades are enabled. This illustrates the importance of having efficient police agents.
3. In some scenarios our agents cannot find alternate routes and get stuck. An area of improvement in our current design is better route planning for the agents.
4. While at the individual level our agents perform well, we need to improve the inter-agent coordination, as we are still significantly less efficient than the competition finalists.

6 Improvements from the previous year

Our police agents did not respond to requests from ambulances and fire brigades. We are adding functionality to allow the police agents to respond to requests from other agents, as follows:

1. Prioritization of emergency requests from ambulance and fire brigade agents. The prioritization is currently based on timestamp and number of times emergency request was made.
2. Allocation of emergency requests based on Sequential Single Item (SSI) auctions. Each police agent bids in auctions for the emergency requests, using its distance from the target as cost for the bid.
3. Proxies for bidding. Since auctions require significant communication, an agent proxy is used in place of a real agent. The agent proxy maintains the status and location of the real agent and is updated every time an agent sends its status to the Police Office.

We will use a new graph partitioning algorithm to divide up the map among the police agents. The police force will start with the full map and divide it successively into two parts as necessary. The algorithm will attempt to divide the map into two pieces of roughly equal size, while minimizing the total lengths of roads spanning multiple partitions. This is to let the police agents to check and clear out any road that is at least partially within their assigned partition, while minimizing redundancy. The implementation is heavily based on the Kernighan-Lin algorithm [7]. It starts by taking the tightest rectangle around all nodes in the map, then finds a line parallel to the shorter sides of the rectangle such that half the nodes land on each side of the line. This creates two preliminary partitions, after which the algorithm attempts to swap nodes and roads between the two partitions to minimize the total redundancy (i.e. the total lengths of roads spanning multiple partitions) while keeping both partitions roughly the same size.

We have improved the ambulance agent performance considerably by implementing a Markov Process overlay on the buildings in the city to use to track civilian locations. Our previous agents only had a very preliminary version of this algorithm - the algorithm has now been fully developed and implemented and has shown very promising results in comparison to other agents. In addition we are working on prioritizing rescue based on the urgency of care required by the different civilians, and working more closely with the fire brigade and police agents to find and rescue them. We have improved the strategies used by the centers and their information sharing, to increase the efficiency with which the agents operate.

We have both improved the strategy of the fire station and the performance of our fire brigade agents. We continue to use k-means clustering to identify distinct groups of fires within the city, but we have improved the cluster definitions and accuracy. Instead of attempting to form convex hulls around clusters of fires, this year we have decided to use a new strategy for assigning and distributing fire agents. We use a weighting system that determines how many agents should be assigned to a cluster, and within that cluster how many agents should be assigned to each building. One of the weighting factors is the wind direction, which the center estimates using the approximate time of ignition and information on buildings on fire. We are also modifying which buildings the fire brigades attend to first, based on their fieriness.

The simulation environment we are dealing is dynamic and partially known. Thus, we need a search algorithm that excels in these types of environments. Up to this point, we were primarily using A* search to plan paths for our fire brigades.Although optimal, this method failed to account for blockade and other path obstruction issues sufficiently. 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 [8] when performing search. It is functionally equivalent to the brute-force replanner, but far more efficient. Its main focus is goal-directed navigation in unknown terrain. Replanning is faster when using D*-Lite, since it is able to modify previous search results locally.

We are presently working to get the agents to work with the new simulator.

7 Conclusions and Future Work

We have presented solutions to research problems in large scale urban search and rescue. We described how we handle coordination in a changing environment, addressing the challenges of task prioritization and allocation among distributed heterogeneous agents. We have shown that agents using our algorithms perform markedly better as compared to the Sample agents.

We have presented a police agent strategy that relies on an approximate solution to the distributed TSP, and have coupled it with a probabilistic update technique to track and rescue civilians, and a cluster-identification system to tackle fires in the city. The performance of the agents has thus far scaled well with increases in the size of the city. The results we have obtained show that our agents perform consistently better than the Sample Agents, and, with the exception of uncontrollable fire spreading, tend to perform well across scenarios with different challenges.

We are working on including multiple enhancements for this year's competition. For example, fire agents will include the wind direction in their decisions about how to move to contain the fire more effectively. We intend to give additional functionality to the centers. For instance, the ambulance center will use a more sophisticated form of book-keeping to improve the update method of the probability distributions maintained over the city, to include civilian health and the urgency of the help needed in various sections of the city. We have implemented an auctioning system in the police agents to handle priority tasks from other agents quickly. Finally, we are working on a strategy to integrate police agents closely with the other two agent types, ensuring passage of information in both directions, since the effectiveness of the team is heavily influenced by the timeliness in which police agents can clear blockades to specific areas.

References

1. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations (1966)
2. Floyd, R.: Algorithm 97: Shortest path. Commun. ACM **5**(6) (1962) 345
3. Warshall, S.: A theorem on boolean matrices. J. ACM **9**(1) (1962) 11–12
4. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms. Prentice Hall of India (2001)
5. Gramam, A., Gupta, A., Karypis, G., Kumar, V.: Introduction to Parallel Computing. Addison Wesley (2003)
6. Graham, R.: An efficient algorithm for determining the convex hull of a finite planar set. Information Processing Letters **1**(4) (1972) 132–133
7. Kernighan, B., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Systems Technical Journal **49** (1970) 291–307
8. Koenig, S., Likhachev, M.: D*lite. In: Proc. Nat'l Conf. on Artificial intelligence, Menlo Park, CA, USA, AAAI (2002) 476–483