MinERS: team formation among heterogeneous agents

James Parker, Ernesto Nunes, Julio Godoy, and Maria Gini
{jparker,enunes,godoy,gini}@cs.umn.edu

University of Minnesota, Minneapolis, MN 55455

Abstract. In order for multi-robot systems to efficiently assist in saving lives and infrastructures in the RoboCup Rescue Simulation, any strategy designed to allocate tasks and coordinate agents must adapt to the dynamic nature of the environment. In this work, we describe how to form teams of agents that take advantage of synergies among the different types of agents and we evaluate the effectiveness of the team configuration strategies on different maps.

1 Introduction

The ultimate goal in natural disasters is to save lives, minimize injuries, and reduce damage to infrastructures. Task allocation has to be done in an environment with uncertainty in the location, type and size of tasks, and requires agents to travel through unknown and possibly unsafe areas to reach the tasks. To make things even more complex, tasks are dynamic, they appear and disappear randomly (e.g., fires). All these elements make urban search and rescue a demanding but realistic testbed.

In our work, we form teams of agents with varying degrees of restrictive behavior using the RoboCup Rescue Simulator. We show that forming teams of agents enhances coordination which, in most cases, increases the number of tasks completed, without increasing communication costs substantially. For the rest of the paper, we first compare various approaches from the literature. Then we explain how we model individual agents and teams, describing how the different types of teamwork are organized and the reasons for the choices we made. We present our results and compare the strengths and weaknesses of the different configurations. Lastly, we offer concluding remarks and ideas for future work.

2 Related Work

In [3], three ways to coordinate the agents are proposed: decentralized mutual adjustment, centralized direct supervision, and environment partitioning. Early approaches to task allocation in RoboCup Rescue were mostly based on centralized coordination, i.e. using a central unit (for example, the Police Station for the police agents) to continuously gather information and assign tasks accordingly. In this category, Sedaghat et al. [6] presented an auction based mechanism

in which all non-police type of agents make requests for blockade clearance to the centralized coordination units, which, in combination with the Police agents, carry a bid for each request and assign a task to the most adequate agent (where the measure for being adequate can be customized). A problem inherent in centralized strategies is the large number of messages required for each task to be assigned to an agent, which makes it unsuitable when a fast response is key to achieve good performance (in terms of the metrics defined by the simulation). Auction based task allocation has been improved in [2], who proposes the use of proxies for agents during the bidding phase to reduce the communication needs, and a collaboration strategy between police and fire brigades. The approach is distributed but still depends on a centralized unit for task requests, which leaves open the possibility for further decentralization. To reduce the single-failure-point issue and improve the robustness of the strategies, more recent work uses decentralized approaches. LA-DCOP, a low communication distributed constraint optimization algorithm, is used by [5]. LA-DCOP is token based and depends on an agent threshold (related to its capability). Each agent decides whether to commit to a task or pass the token to more suitable agents. Ferreira et al. [1] compare LA-DCOP with Swarm-GAP, which emulates insect behavior to achieve implicit coordination. In [4] task allocation is addressed with coalition formation, solving the problem as a distributed optimization problem using Max-Sum, and including spatial and temporal constraints. However, the approach is not evaluated in RoboCup Rescue.

3 Agent Description

3.1 Abstract agent

The abstract agent is a template we use to build the three agent types needed for RoboCup Rescue. All agents inform others when they cannot move because of blocked roads, which buildings are on fire, and where buried civilians are located. However, due to communication limitations, only fire brigades are informed of buildings on fire and only ambulances are informed of buried civilians. For path planning, we use A* with a slight modification: roads that are impassable due to blockades are given a very high cost to expand them in the search (we use the normal total cost multiplied by $200,000$). This ensures that any reasonable unblocked path will be explored first and if all paths are blocked, the shortest path with fewest blocked roads will be picked. A road is classified as impassable if the smallest choke in the road is smaller than the size of an agent, which means an agent cannot fit through this part of the road.

3.2 Fire Brigade

Fire brigades assign to target buildings a value which is inversely proportional to the distance and fieriness of fire:

$$Val_t = \frac{100}{D_t \times F_t},$$

where Val_t is the value of a target t, D_t is the distance of the agent to target t and F_t is the fieriness of the target building t. Fires that have grown large have a large fieriness and are much more difficult to put out, so by using this value fire brigades will attempt to put out close fires that are small. Since the values for distance are much larger than the one for fieriness, distance has a larger impact on the decay of the value. Buildings that are not on fire are not assigned a value and are not considered a target.

Once the most valuable target is identified, the fire brigade will head towards the target building, and as soon as the building is in sight it will start extinguishing the fire. Fire brigades constantly look for best targets, so if a more valuable target appeared the fire brigade would immediately switch to it. An example is when fire spreads to a nearby building, fire brigades will quickly put out the new small fire before returning to the old one to ensure the fire remains contained. The fire brigade returns immediately to the refuge to refill its water tank when empty.

3.3 Ambulance Team

Ambulance teams select buried civilians as targets based on the value:

$$Val_t = \frac{100}{D_t} - B_t,$$

where B_t is the degree of buriedness of the target t, and the other variables are the same as for the fire brigades. Larger values of B_t mean the civilian is buried under more rubble and will take longer to be uncovered. The values for distance are again much larger than the ones for buriedness, so ambulance teams will generally select the easiest to uncover target from the closest cluster of targets. This maximizes the number of civilians who will be saved if the building catches fire and ambulance needs to vacate. This also coordinates the effort of all ambulances working on a cluster to uncover the same target. When a target is unburied, one ambulance will load up the target and take it to a refuge and the rest will go to the next most valuable target. Ambulance teams will select new targets if a closer trapped target is spotted or reported by another agent.

3.4 Police Force

Police forces choose blocked roads as targets based on the value:

$$Val_t = \frac{100 \times I_t}{D_t},$$

where I_t is 1 if the road is completely impassable and decreases rapidly towards zero, but never reaching, as the size of the smallest choke increases. This makes police forces select roads that are impassable, even if far away, before selecting roads that are passable but with suboptimal throughput. Once a road is selected as a target, the police will travel to this road and clear all the blockades on the road to make it passable again.

4 Teamwork Configurations

In this section we describe different methods we developed for teamwork. Specifically, we describe the case where no team work is done (which we call "base"), the case where teams are created at the start of the simulation and do not change over the run (called "static"), the case where teams change as required by the tasks (called "dynamic"), and the case where the teams are split to have some teams with ambulances and some with fire brigades (called "split").

4.1 No teamwork

When no teams are formed each agent acts independently as described in Section 3 and since there is no coordination the centers do nothing. Agents freely select the target t with the highest Val_t as described in Section 3. This shows the benefit from communicating and selecting targets by value over the sample agent. No teamwork, called base configuration, is also the baseline we use to compare the different teamwork strategies against.

4.2 Static team assignments

In the static team configuration, teams are formed at the start of the simulator based on a bottom-up hierarchical clustering algorithm and do not change after that. Each agent starts as its own cluster with the position of the agent in the simulator. The closest pair of clusters, based on the Euclidean distance between them, are merged and the mean position of all agents in both clusters is then assigned as the new cluster's position. If a merge creates a resulting cluster of over 7 agents, then the larger cluster before the merge is considered a finished team and is neglected from the rest of the clustering algorithm. This process is repeated until all teams are finished or if the distance between all clusters is too far (1,000 meters in the simulation, about five blocks). Teams are formed in this fashion to ensure spatial locality of the agents on the team. To implicitly regulate agents on a team to work together, an area is formed which all agents on that team are expected to operate in. The center of the team is ($\bar{x}$,$\bar{y}$) defined by:

$$\bar{x} = \frac{\sum_i x_i}{\sum_i 1}, \bar{y} = \frac{\sum_i y_i}{\sum_i 1},$$

where x_i and y_i are the x and y coordinates of agent i, and the sum is over all agents on the current team. The team area is then approximately a circle of radius two blocks (400 meters in the simulator) around ($\bar{x}$,$\bar{y}$), and each agent chooses the target with maximum $TeamVal_t$ within this area as described below. The two block distance is a heuristic used that allows a good balance between having enough area to find suitable targets and being small enough to force implicit cooperation.

Centers are used to keep track of where these team areas are located throughout the simulation. Agents send their individual positions to the centers, who

then calculate the average position of the team and send it back. If team updates are too infrequent, the center of the team moves very slowly and not much of the city can be visited by the team. Frequent updates can cause target thrashing, where an agent moves to a target but the target is outside the team area in the next update. This forces the agent to select a new target and waste more time in transit. To help reduce this problem Val_t is modified by a sigmoid function $P(t)$ to devalue targets near the edge of the team radius:

$$TeamVal_t = Val_t * P(TD_t/TR),$$

where Val_t is as described in Section 3, TD_t is the distance from $(\bar{x},\bar{y})$ to target t and the $TR = 400$ meters as the team radius described above. Team updates are staggered and sent periodically (every 5 time steps in the simulator) to provide low communication requirements and avoid target thrashing.

4.3 Dynamic team assignments

Unlike static teams where agents are assigned to the same team for the whole simulation, dynamic teams allow agents to change from one team to another. To accomplish this each agent is assigned a *Utility* as an approximation to the effectiveness of that agent on the team. The *Utility* is increased by 4 when interacting with a target, decreased by 1 when moving to a target or refuge, and decreased by 2 if moving randomly looking for a new targets. To balance longterm benefits with short term benefits, *Utility* is limited to the range between zero and 100. This *Utility* is sent to the centers along with the normal periodic team updates described in Section 4.2. The center then calculates three separate average utilities per team, one for each of the agent types. If an agent type is not on the team, the utility is set to 50. These averages are calculated separately since some agent types may have very high utility and others very low utility. A short-term expected utility gain method is used to determine whether it will be beneficial for an agent to move to a different team. We defined short-term as a time window of $t = 10$ simulation time steps, then the expected utility gain of going from team j to team i would be:

$$Gain_{i,j} = U_i \times (t - TravelTime_{i,j}) - U_j \times t,$$

where U_i is the average utility of the agent on team i and $TravelTime_{i,j}$ is the expected time for an agent to travel from team j to team i. The best team b for an agent on team j to move to is then: $Gain_{b,j} = max_i Gain_{i,j}$. If $Gain_{b,j}$ is positive then one agent of that type with the lowest *Utility* on team j is transfered to team b, via a message by the center. Only one calculation for each agent type on a team occurs when that team is sending update messages.

4.4 Split ambulance teams and fire brigades

One drawback of the teamwork is that fire brigades and ambulance teams often have conflicting goals. Both extinguishing fires and unburying civilians are

time consuming tasks, unless the fire is just starting. This can frequently cause agents of one type to have a very high utility while others have a very low utility. Since police forces do not directly affect the score, they play a secondary role by allowing the fire brigades and ambulance teams to do their tasks efficiently. To maximize the primary agent's utility, teams are not allowed to have both ambulance teams and fire brigade teams on them. We call this the split configuration. When ambulance teams and fire brigades are on different teams, the hierarchical clustering is done in two repeated stages. In the first stage only fire brigade and police force clusters are considered, and the closest two clusters are merged as described in Section 4.2. If this merger was between a fire brigade cluster and a police force cluster it is considered a fire brigade cluster, and the old police force cluster must be removed from the pool of clusters available to the second stage. The second stage considers only ambulance teams and police force clusters, and again the closest two cluster are merged. Similarly if this resulting merger has heterogeneous agent types, the cluster is considered an ambulance cluster and any police in that cluster are unavailable to stage one. This round-robin hierarchical clustering ensures that teams will be formed based on local proximity while not allowing a team to have both fire brigade and ambulance team agents. Utility is still tracked in the split configuration and agents can move between teams, however fire brigades cannot move to a team with an ambulance and vice versa. To implement this, when centers compute the expected utility gains to transfer agents, the gains are only evaluated between teams of similar types for the fire brigade and ambulance team agents. This is done by looking at whether the team was a fire brigade or ambulance team during the initial formation. Police forces are still free to transfer to any cluster and follow the dynamic team configuration outlined in Section 4.3.

5 Results

5.1 Overall analysis

The maps and their configurations from Table 1 are from the 2011 RoboCup Rescue Simulation League final round. As seen in Table 1 the split configuration outperforms the others. The best configuration for each map is shown in bold. The dynamic configuration does slightly worse than split, but better than

Table 1. Scores obtained by the different teaming configurations on various maps.

	Sample	Base	Static	Dynamic	Split
Paris	1.57	0	0.35	1.15	**2.25**
Berlin	18.37	**30.27**	21.92	21.41	26.77
Istanbul	28.68	12.62	11.56	55.25	**58.49**
Kobe	64.98	54.79	54.79	76.70	**81.66**
VC	21.54	11.77	11.42	80.59	**93.24**
Total	135.24	109.46	100.06	235.10	**262.41**

the base and static configurations. This shows that forming teams for task cooperation is helpful in this time sensitive search and rescue problem. Not only does teamwork increase the efficiency of agents, but even without much specific tweaking to the simulator, the split agent configuration places close to 3rd place of the final round competitors.

On Paris, the split configuration is significantly different from the static and base configurations based off a two sample t-test with 95% confidence. Unfortunately since the scores are low on this map , saving an extra civilian causes a large percentage increase in score. This results in a high standard deviation for most configurations making conclusive results about the means difficult. The split vs. base and sample vs. dynamic configurations were not significantly different on Berlin, but all other pairings are different with 95% confidence. For Istanbul, Kobe and VC only the base and static configurations were indistinguishable due to the very similar estimated means.

Static teams do not do well because agents are not used efficiently. One strength of static teams is attempting to rescue civilians near fires since the fire brigades can keep the burning buildings away from ambulances long enough to rescue civilians. The problem with this approach is that many fire brigades are on teams with ambulance agents rescuing civilians not near fires. This inefficiency of the fire brigades causes a vast portion of the city to burn down where teams are not present and burns many civilians before agents become aware of their existence.

Dynamic teams help overcome this inefficiency of agents by allowing them to move to a team where they will be utilized more. Teams tend to accumulate one type of agent and lose some of the opposite type. The police force are the agent type that changes team the most frequently, due to their faster task completion rate. Fire brigades can spend the entire simulation trying to contain a raging fire, but the police will run out of roads to clear quickly. These unused police agents will switch to teams that are still mobile and encountering new impassable roads. The ability to switch teams substantially increases the average utility of agents and as a result the dynamic configuration always performs better than the static configuration.

The overall best configuration is the split configuration, which is inspired by seeing a disparity in the efficiency of agents on the same team. This configuration still retains the dynamic aspect allowing all agents to optimize their utility.

6 Conclusions and Future Work

We have shown that creating teams can exploit synergy between heterogeneous agents and creates implicit task coordination. Restricting agents on a team to a local area increases cooperation among agents. Additionally, since individual agents have similar criteria for choosing targets and they are restricted to a local area they will often select the same target producing coordinated effort among them. We apply four different team formation strategies: no teamwork, static teams, dynamic teams and separated ambulance and fire brigades. We compare

the various strengths and weaknesses of our configurations against the baseline of the sample agent. Dynamic teams outperform no teamwork and the split agent configuration outperforms dynamic teams in turn.

Work remains to be done on exploring team benefits when agents are able to accomplish all tasks but have different specializations. Our best configuration is the separated ambulance teams and fire brigades because police forces play a supporting role. In the simulator an obvious choice is to split the teams up in this manner, but how can team compositions be automatically learned depending on agent abilities remains an open question. The choice of team radius and the time window selection for changing team could have an impact on performance and deserve additional investigation.

One of the benefits to having teams is the ability to react quickly to new information as it arises. However, as the time progresses and the maps are explored, new information becomes less frequent and this benefit is lost. At some point agents become more effective if they work independently instead of assisting other agents. This behavior could be treated as an explorations vs. exploitation type problem, but it is unknown if there is a accurate model for various situations.

References

1. P. R. Ferreira, Jr., F. dos Santos, A. L. C. Bazzan, D. Epstein, and S. J. Waskow. RoboCup Rescue as multiagent task allocation among teams: experiments with task interdependencies. *Journal of Autonomous Agents and Multi-Agent Systems*, 20(3):421–443, 2010.
2. M. Nanjanath, A. Erlandson, S. Andrist, A. Ragipindi, A. Mohammed, A. Sharma, and M. Gini. Decision and coordination strategies for robocup rescue agents. In *Proc. SIMPAR*, pages 473–484, 2010.
3. S. Paquet, N. Bernier, and B. Chaib-draa. Comparison of different coordination strategies for the RoboCup rescue simulation. In *Proc. Int'l Conf. on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems*, volume LNAI 3029, pages 987–996. Springer-Verlag, 2004.
4. S. Ramchurn, A. Farinelli, K. Macarthur, M. Polukarov, and N. Jennings. Decentralised coordination in RoboCup Rescue. *The Computer Journal*, 53(9):1–15, January 2010.
5. P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Allocating tasks in extreme teams. In *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, pages 727–734, 2005.
6. M. N. Sedaghat, L. P. Nejad, S. Iravanian, and E. Rafiee. Task allocation for the police force agents in RoboCup Rescue simulation. In *LNAI*, volume 4020, pages 656–664. Springer, 2006.