RoboAKUT 2010 Rescue Simulation League Agent Team Description

H. Levent Akın, Orçun Yılmaz, and Mehmet Murat Sevim

Department of Computer Engineering
Boğaziçi University, 34342, Istanbul, Turkey
{akin,orcun.yilmaz,mehmet.sevim}@boun.edu.tr
http://robot.cmpe.boun.edu.tr/rescue

Abstract. RoboAKUT is a multi-agent rescue team developed at the Artificial Intelligence Lab of the Computer Engineering Department of Bogazici University. Our primary goal is to build a rescue team based on the market paradigm with the mixture of regional task management, then show that this method can be successfully implemented in highly dynamic, multitasking and multi-robot environments. In this paper, we give a detailed description of the software architecture and the algorithms to be used by the RoboAKUT team. Since in 2010 a new simulator and a new library are released, we have developed new agents for the competition. New rescue strategies are implemented together with the strategies of the previous year. In addition, exploration task is achieved in a more effective manner by dividing the world into regions. Enhanced estimation mechanism, noise handling, search algorithms and communication layer are the other main improvements over RoboAKUT 2009.

1 Introduction

RoboCup Rescue Simulation environment is a disaster management simulation which consists of multi-tasking heterogeneous agents (Fire brigades, Fire Station, Police Forces, Police Office, Ambulance Teams and Ambulance Center). In addition to being one of the best test beds for agent coordination, there are many other challenges such as development of agent communication protocols for limited communication and noisy message arrivals, multi-agent path planning, scheduling, optimization, supervised learning for civilian death time and fire behavior estimation and unsupervised learning for agents to develop policies.

RoboAKUT is a multi-agent rescue team developed at the Artificial Intelligence Laboratory of the Department of Computer Engineering of Bogazici University. Team performs rescue operations on the simulation environment provided by the RoboCup Rescue Simulation League. RoboAKUT has participated in the RoboCup Rescue Simulation Competitions held in Fukuoka, Japan in 2002, RoboCup 2003 held in Padua, Italy, RoboCup 2004 in Lisbon, Portugal, RoboCup 2005 in Osaka, Japan, RoboCup 2006 in Bremen, Germany, RoboCup 2007 in Atlanta, USA, and RoboCup2008 in Suzhou, China, RoboCup 2009 in Graz, Austria.

Version 1.0 of the RoboCup Rescue Simulation League platform has been released for RoboCup 2010. This version of the simulator has many changes including a new

library for developing rescue agents. For this year, RoboAKUT agents that use the new *Rescucore2* library are implemented.

In this team description paper, the team members and their contributions are presented at the next section. The key contributions of our team are briefly explained in the second section. At the software architecture section, package architecture of RoboAKUT agents are described. World model section presents algorithms for dividing world into regions and assigning them to the agents. In addition, communication section explains the structure of the communication layer which includes priority mechanism for message passing and noise reduction technique. Estimation section talks about mechanisms which are used for estimating civilian life span and fire state. Rescue strategies, map regions and market algorithms are presented at the agents section. And as a result, the last section shows our conclusions for the project.

2 Team Members and Their Contributions

- Mehmet Murat Sevim (Developer)
- Orçun Yılmaz(Developer)
- H. Levent Akın (Advisor)

3 Key Contributions and Improvements Over 2009

The architecture of RoboAKUT is implemented from scratch for the 2010 competition. The most important key contributions and improvements are the following:

- **RoboAKUT world model** is a new world model which divides the map into regions. The number of regions are determined from the number of agents in the map. The main roads of the city are used to determine the borders of the regions. In order to visualize the regions a special viewer has been implemented. This viewer shows the regions and assigns different colors to the regions and the main roads of the city. The RoboAKUT world model is explained in more detail in Section 5.
- **Communication System** is basically a middle layer between the agents and the kernel to send/receive messages. This layer can be considered as a filter for each agent to get involved with the messages which they are related to. Since communication is the most important factor in multi-tasking and multi-robot environments, we developed a technique for noise reduction in communication. Moreover, we developed a priority queue structure for our communication layer in this environment where communication is a scarce resource. This structure is explained in more detail in Section 6.
- **Estimation Mechanism** is used to estimate the conditions of the civilians, the buildings and the roads on the map in order to assign the agents to tasks more appropriately. This system includes fire state estimation and civilian life span estimation.
- **Task Assignment Mechanism** is a novel approach that is peculiar to RoboAKUT. In addition to the regional task assignment system, an auction system is being used for the task assignment to each agent. This creates a hybrid market paradigm and regional task assignment system. Detailed information is included in Section 9.

4 Software Architecture

Our software system is based on the *rescuecore2* library. The main modules are:

- **communication :** provides enhanced priority management and message queuing functionalities. All types agents have specialized communicator objects,
- **communication.message :** includes the message types,
- **decision :** contains decision manager class for the rescue agents,
- **decision.task :** implements task classes for agents,
- **standard :** contains extended versions of entities and world model,
- **standard.region :** manages the regions of the world model,
- **search :** provides search algorithms for the map,
- **roboakut.viewer :** provides viewer for world model,
- **roboakut.market :** implements market algorithms to the rescue environment,
- **roboakut.agent :** contains implementation of RoboAKUT agents.

5 World Model

A basic world model is provided by the simulator. This model keeps the minimum required information about the world. A special model which is an extended version has been implemented for the RoboAKUT agents. Classes of some entities in the world are also extended as they are needed.

Fig. 1. A scene of divided regions from RoboAKUT viewer

The most important addition to the model is the regions as shown in Fig. 1. In order to keep the exploration and tasks organized, the map should be divided into regions and the agents should be assigned to these regions. Each agent is responsible for the tasks in certain regions and the agents may also help the other agents when needed.

The separation of regions is done in the first cycle of the simulation by every agent. The agents then determine the region they are currently in and tell the center that they want to be responsible for this region. The center handles conflicts and unassigned regions, and then sends the new region distribution to the agents. At the beginning of the third cycle all the regions are set and they stay the same throughout the entire simulation.

Moving agents within regions and moving agents from one region to another should be done in an efficient way. Two buildings in a single region should be close to each other and should be connected with short paths. Therefore, how regions are determined is important. In order to determine the regions, first, the main roads in the map are detected. They are the sequence of roads in the map that have the most lanes. After the detection of the main roads, the regions are determined as the areas that are between the main roads.

6 Communication

Communication is considered to be the most important part of our system. A special layer is implemented for handling communication. The agent just tells the content, priority and the channel of the message and all details such as how the message is compressed or which method is used to send the message is determined by the communication layer.

The total length of messages sent in a channel is limited. In order to send messages effectively, a priority queue of messages is implemented. The agent tells the message and its priority to the communication layer. The agent may tell more than one message. All messages are added to the priority queue. At the end of the turn, the messages with the highest priority are selected. The number of selected messages depends on the length of the messages and the bandwidth limit. The messages are selected such that the total length of the messages does not exceed the bandwidth. Then the selected messages are sent to the kernel. The remaining messages that are left in the queue may be deleted depending on their validity. If they are not deleted their priorities are updated in the next turn.

6.1 Messages

A special message architecture is implemented to make it easier to add new message types and modify the existing ones. Each message has a list of properties that consists of message type, message id, and type specific properties. The data type and the number of bytes required for storing is known for each property. The messages are converted to bytes with this information. Conversion is independent of the message type and no extra implementation is necessary to convert specific messages to byte arrays.

The message type is used to convert the bytes to the associated message object. The id is used to prevent the agents from processing a message more than once. The total number of bytes that are used in a message representation is the sum of bytes required for properties. For example, a ”Building is explored” message contains only the id of the building and two bytes are sufficient to store this property. With the addition of two bytes for message type and id, four bytes are enough to represent this message.

6.2 Noise Handling

One of the challenges about the communication is noise. The static noise is removed from the latest rules so there are only two types of noise, namely *dropout* and *failure*. We handle both types of noise by sending the message more than once. The number of times a message is repeated depends on the probability of noise on the radio channel. The noise levels of channels are different than one another so different number of copies are used for different channels. On a channel without noise the messages are sent only once.

The following formula is used to determine how many copies are required for a channel.

$$numberOfCopies = \lceil (\frac{log(0.001)}{log(p)}) \rceil$$

where p indicates the sum of probabilities of noise in the channel. This formula assures that 99.9 percent of the messages are successfully received by the recipients. Note that, in most cases the recipient receives multiple copies of the message but a message is not processed more than once since the same messages are detected by ids and extra copies are ignored.

7 Search Algorithms

A* search is used for path finding. The implementation of RoboAKUT 2009 is ported to the new platform. In a disaster environment finding the best path depends on two main factors. These factors are the length of the path and the blocks on the path. One path may seem efficient in terms of distance, but it may contain a lot of blocks and clearing these blocks may require many simulation cycles. As an extension for the search algorithm of previous year, blocks on the path are also taken into account when determining the best path.

8 Estimation Mechanisms

8.1 Fire Estimation

Each fire brigade agent holds a fire risk map of the city. This map is updated in each cycle of the simulation according to available new data. This risk map directly maps building ids and the estimated properties of these buildings. Fire risk property for a

building is calculated from fieriness of the nearby buildings and the ignition of the building.

The fire brigades check the total risks of a fire site continuously. They determine how many fire brigades are needed for the fire site by using the sum of the fire risks of the buildings on the site.

In order to determine the costs of extinguishing a building task, how much simulation cycle is needed to extinguish must be estimated. This estimation takes also the necessary water refills into account. Moreover, it also considers fieriness and the total area of the burning building.

8.2 Civilian Health Estimation

A civilian is categorized as *helpless, will die soon, will die* and *will live* according to estimated time to live and closeness to a disaster area. In order to determine the category and costs of saving the civilian, how much simulation cycle is needed must also be estimated. If an ambulance team agent finds a buried civilian it can calculate this value. However, if another agent explores a buried civilian and informs an ambulance team, the ambulance team agents should consider the path to reach the civilian in addition to the saving time estimate. This path most probably will contain some blocks and the number of blocks is not accurate. Reaching the rescue site is just the first part for this task. Costs of reaching to a refuge later is also considered.

For the auction mechanism, the cost of doing this job is estimated by taking the predicted road blocks into account. This cost is very high for the farthest ambulance team agents. Therefore, in most cases the nearest agents will win the task. This shows an advantage of using the market paradigm in task assignment.

9 Agents

9.1 Map Regions for Agents

As described in Section 5, the whole map is divided into regions and the regions are assigned to the agents such that every region has an agent of each type assigned to it. The regions also have an important role in the auctions. Every task has an associated region and the auction for the task is lead by the agent responsible for that region. The type of the agent depends on the type of the task. The agent may choose not to lead the auction and it can do the task itself if the profit of the task is greater that its current task. In this case, the agent starts another auction to let the other agents complete its current task.

The exploration of the map significantly depends on the regions. The exploration tasks are not auctioned and they are handled by the agents of the associated region. The exploration may be interrupted by other tasks as the agents take new tasks. However, the map should be explored as soon as possible. In order to achieve this, the reward for exploration is increased in the later steps of the simulation.

9.2 Multi-agent Coordination

Although multi-agent systems have been developed for solving different types of problems, these problems share some common characteristics. Gerkey and Mataric [3] developed a taxonomy based on three attributes of the problem definition. Here after we will refer to this taxonomy as GM taxonomy. First, the problem is categorized as either single robot (SR) or multi robot (MR) depending on whether the task can be achieved by one or more robots. The next categorization is done regarding whether a robot is capable of only a single task (ST) or more (MT). Finally, the problem is categorized as instantaneous (IA) if all the tasks are known by the agents initially. The counterpart definition of the assignment property is time extended (TA) which describes the problems where the tasks are discovered during the course of action. These definitions are useful in classifying the problems in the multi-agent domain.

The RoboCup Rescue Simulation domain contains many complex problems and is therefore very suitable for investigating multi-agent team development approaches. The agents are clearly heterogeneous because according to the rules, only police forces can clear roads, only ambulance teams excavate and transfer civilians and only fire brigades can extinguish fires. The agents are MT agents where each agent is capable of more than one task with tasks to its type and in addition, common tasks like finding a civilian, watching fires and exploring the map. Some of the tasks are MR because of the resource constraints, e.g., an ambulance team cannot rescue a civilian on time alone, a fire brigade cannot extinguish a fire alone, etc. Moreover, since the roads are blocked, any task containing a reach target task is MR because the roads must be cleared first.

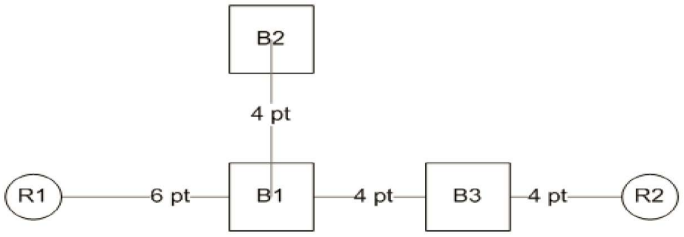

Fig. 2. A task allocation scenario

Both types of task assignment are available in the system, since the map is known by the agents. Exploring the whole map is a IA type of task and since the blocked roads, fires and civilians are discovered during the simulation the tasks related to them are TA tasks. It is also possible to simulate other types of problem domains by applying different configurations to the system. Single item auctions are not sufficient for ensuring system optimality. Single item exchanges between robots with or without money lead to poor and suboptimal solutions in some but apparently possible cases. A simple scenario is depicted in Figure 2. The robots auction for the tasks which costs the lowest for them. Therefore after the first auction round R1 adds B1 to its task list and R2 adds B3 to its task list. In the second round R1 will add B2 because it becomes the best alternative for

the task. After all the targets are finished B2 will not be exchanged because it is best handled after B1. The auction rounds are finished when all the agents are assigned to the tasks. But after this, single exchanges cannot remedy the problem. The solution is to allow some of the agents not to be assigned to any task as our work suggests.

We propose that by taking a simple plan into account while bidding in auctions, the agent is capable of exchanging multiple items in single item auctions. The proposed approach [2]is implemented mainly in the multi-robot exploration task where centralized solutions do not satisfy the communication and robustness requirements. In such solutions, all the agents communicate with the center which introduces the single point of failure problem. Moreover, if communication with the center fails or is noisy, the performance degrades sharply even to non-functioning level. The exploration task, however, must be completed in any condition even when only one robot survives.

In our approach, the agent simply plans a route that covers all the known targets by using the TSP insertion heuristic. Each agent auctions for the lowest cost target which is in fact the closest target to the agent. Other agents bid in the auction according to their plan cost. The plan cost is the distance between the agent and target if the target is the closest target or the distance to the previous target in the plan. The pseudo code for the algorithm is given in Figure 3.

```
check whether the target is reached
plan current tasks
bid for the lowest cost item
if auction won
        allocate task
else if  deadlock detected,
        solve according to total plan cost
else
        stay
```

Fig. 3. Market plan algorithm pseudo code

Task Representation

Task definition is the key for a successful framework because it forms the basis of agent abilities in the market. The properties of a task in our framework are discussed in the following sections.

Task Constraints. There are several resource constraints in a system due to the fact that there are many different objectives to maximize utility of many resources such as time and fuel used by the agents. For the case of heterogeneous robots, robot types and capabilities are also a kind of resource constraint. In addition to the resources of a multi-agent system there are also coordination and synchronization constraints for tasks. The tasks may depend on each other's completion, the start and/or end time of a task. We propose task decomposition as a graph that represents these properties. Moreover we define task milestones for even atomic tasks when necessary. Task decomposition is used by police force agents to sell their tasks to other agents to minimize the total time of path clearance.

Task Milestones A task can depend on the completion of another task. Therefore task dependencies prevent the agents from starting a task before the depended tasks are totally completed. This preventing the start of a task is frequently encountered. For example, assume that one of the agents is occupying a grid and will use it for some time. Another agent, however, needs to use this grid to pass, but it knows that it is occupied for some time. In this case, the agent should be allowed to move along the path at least up to a destination which we call milestone. There are different kinds of milestones during the course of the task and the start is only one of them.

Task Opportunities Since the plans consist of the best task orderings, we can say that they eliminate the need for combinatorial auctions and put a kind of agreement which is calculated to be the optimum way of doing the job. However the IT cases cannot be handled because these domains do not allow future planning of the tasks because of the dynamic environment. The problem can be solved by combinatorial auctions if it is ST-SR-TA or ST-SR-IA and task decomposition if it is a ST-SR-TA.

In our approach, for each task type, we propose to define a list of possible task types that can be executed simultaneously. Before bidding, the agent checks its task list and tries to find a task opportunity that can increase its revenue, i.e. decrease the cost of taking a task. The benefit of this approach is the same as task decomposition which is trading for a task where the combination of tasks is planned behind the scenes and implementing such a scheme without using complex task models in the framework.

The default task opportunity for a task type is itself. Therefore task opportunities are the common version of the plan based bids paradigm. However its implementation is easier since each task type is encapsulated and complexity is reduced. For example, a police force can remove other blockades on its path to a specific blockade target. In contrast, an agent can explore buildings on the way to the blockade and while bidding it can offer less cost to the system.

Cost and Revenue Each task has a cost based on the resource requirements. However having representations that depend on only the cost is not sufficient for multitasking environments, since an agent may minimize its costs but this does not yield maximization of the overall profit. Therefore the revenue for each task must also be defined and used by the system. In a market economy every asset has its price and the price is determined by the market mechanisms like supply and demand. So the price is not static and it is also subject to change according to the changes in the environment. Therefore revenue is not static. In multitasking dynamic environments where tasks are assigned in time extended fashion any task can be abandoned if its cost is affordable. As a result, the cost is not only a function of the resource usage but also the opportunities that the agent is missing.

Task Notifications All the tasks the agent can execute or can be a part of are stored by all agents. Therefore no task is left unexecuted because the task is never lost due to robot failures or communication errors. But the agents cannot store the tasks forever even if the task is finished or expired. The task status also needs to be synchronized. Task notifications are defined for each task and sent when necessary.

Task Execution Preconditions of a task is valid for some time frame, a world state or simply exists in the environment. These preconditions must be checked for each time step. If an inconsistency is discovered other agents are notified as described above.

Task Allocation and Coalition Formation

Coalitions are necessary for most of the heterogeneous agent domains. Because of the diversity of the robot capabilities, typically one agent is not capable of accomplishing a task alone. Tasks can be categorized as ST when one agent is able to achieve it or MT when a coalition needs to be formed.

- ST case: The task is auctioned and one of the agents wins the auction. After this, if the task is decomposable the winner agent starts auctions for sub-tasks. A coalition is formed if any part of the task is sold.
- MT case: The tasks are auctioned depending on the task graph and a single task is reached. Then the subtasks are treated like the single task case.

Both cases are almost identical and show that the tasks can be decomposed and auctioned iteratively. For ST tasks there is another case in which the coalition emerges from the needs of the heterogeneous agent teams. The task types are predefined as ST or MT but while executing, a ST task that must be done by other types of agents can be discovered that blocks the agent and prevents it from doing the task. For example, while trying to reach a target, a block can be discovered on the road and unluckily on the only path to the target. In this case, the agent requests help from other agents by auctioning as in the two cases. We call this dynamic coalition formation and the required task is added to the task graph of the requiring task.

10 Results

We have tested the current implementation with the latest working simulator package using the Kobe map as shown in Figure 4. In the sample runs about 60 percent of the civilians are rescued. The total health proportion of the civilians at the end of the simulation is nearly 0.5 as illustrated in Figure 5. The fires are extinguished before they spread and in most cases there are only a few collapsed buildings at the end of the simulation. Building damage score is more than 95 per cent in all runs.

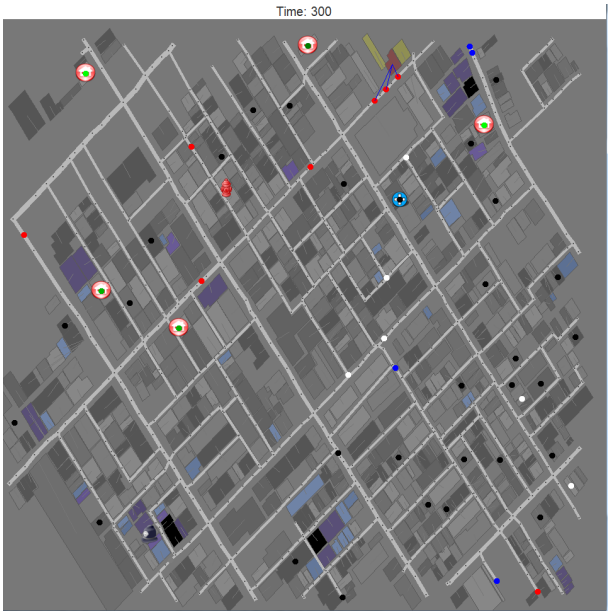

Fig. 4. The world at the end of the simulation

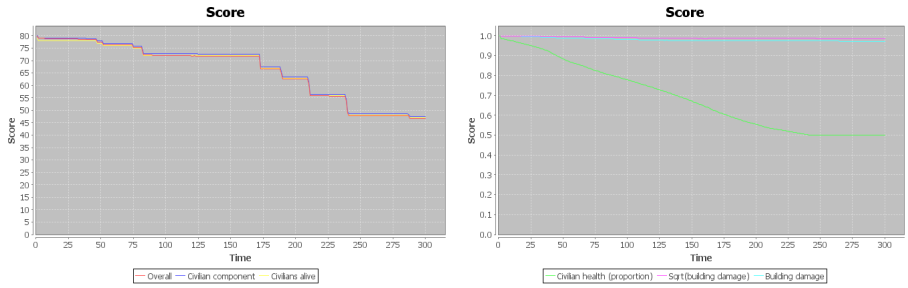

Fig. 5. The score graphs of a sample simulation

11 Conclusion

In this paper we presented an overview of the agent system model and the algorithms of RoboAKUT 2010 team which consists of market based agents which can utilize resources effectively even under dynamic conditions. The exploration of the map has been optimized by using a world map with regions. Enhanced estimation mechanism, noise

handling, search algorithms and communication layer are the other main improvements over RoboAKUT 2009.

The test runs on the new simulator show that the fires are successfully extinguished and the majority of the civilians are saved.

References

1. Kose, H., K. Kaplan, Ç. Meriçli, U. Tatlıdede, and L. Akın, "Market-Driven Multi-Agent Collaboration in Robot Soccer Domain," in V. Kordic, A. Lazinica and M. Merdan (Eds.), *Cutting Edge Robotics*, pp.407-416, pIV pro literatur Verlag, 2005.
2. Tatlıdede, M. U. and H. L. Akın, "Planning for Bidding in Single Item Auctions," First International Workshop on, Agent Technology for Disaster Management (ATDM), pp.85-90, 2006.
3. Gerkey, B. and M. Mataric. "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of Robotic Research*, vol. 23, pp.939–954, 2004.