RoboAKUT 2015 Rescue Simulation League Agent Team Description

H. Levent Akın and Okan Aşık

Department of Computer Engineering
Boğaziçi University, 34342, Istanbul, Turkey
{akin}@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 Boğaziçi University. In this paper, we give a brief description of the software architecture and the algorithms used by the RoboAKUT 2015 team. Our current rescue team is based on the market paradigm with regional task management. For the RoboCup 2015 competition, based on our experiences in RoboCup 2014 we made some changes to improve the effectiveness and robustness of the developed algorithms. Additionally, we are now in the process of implementing a Decentralized Partially Observable Markov Decision Process based approach. We are working on a new layered Dec-POMDP approach for solving Dec-POMDP problems having big action space. Our results with the RoboCup 2014 maps show that the RoboAKUT 2015 team will be a serious contender in RoboCup 2015.

1 Introduction

RoboCup Rescue Simulation (RSL) agent competition consists of a disaster management simulation with multi-tasking heterogeneous agents, i.e.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, it contains many other challenges such as the 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 Boğaziçi University. The team performs rescue operations on the simulation environment provided by the RoboCup Rescue Simulation League. RoboAKUT has been participating in the RoboCup Rescue Simulation Competitions since 2002. Our team has won the **First Place** in the agent competition in RoboCup 2010 Singapore.

The rest of the paper is organized as follows. In Section 2 the team members and their duties are introduced. The contributions made for RoboCup 2015 are given in Section 3. In Section 4 the Market algorithm and its implementation for the fire fighters, ambulances and police forces are described. The layered Decentralized Partially Observable Markov Decision Process (Dec-POMDP) based approach is described in 5.

The current scores of RoboAKUT 2015 are given in Section 6. The conclusions are given in Section 7.

2 Team Members and Their Contributions

- Okan Aşık (Developer)
- H. Levent Akın (Advisor)

3 RoboAKUT 2015

RoboAKUT team code was completely rewritten for the RoboCup 2010 competition [1], and has been revised each year since. RoboAKUT 2015 is the optimized and improved version of the RoboCup 2014 [2] code based on the competition experiences.

RoboAKUT 2015 currently uses subsumption based architecture [3] and a market driven approach [4] in the task assignments to the agents. In addition to the regional task assignment system, we use an auction system for the task assignment to each agent. This is an effective hybrid market paradigm and regional task assignment system. We are now in the process of adapting a new layered approach based on Decentralized Partially Observable Markov Decision Process (Dec-POMDP) developed in [5, 6] and which was successfully applied to robot soccer [7].

3.1 Agent Architecture

Our agent architecture has the following basic components; distributed world modeling, priority based communication and subsumption behavior component. The agent first creates a world model where it keeps information about the other agents, traveled buildings, civilians. Then, the agent uses this world model to choose a behavior from the subsumption behavior component. In the subsumption architecture, the behaviors are designed as independent entities. Each behavior has a certain priority over each other and certain activation conditions. The final behavior of the agent is chosen according to priority and activation condition evaluated according to the world model of the agent.

Distributed World Modeling The first step of the world modeling is dividing the map into regions. Currently, we divide the map into equal sized grids. We determine the number of grids according to the number of agents. Finally, we cluster regions according to building area for fire brigades, road count for police force, and building count for ambulance. The agent closer to the center of a region cluster is assigned to this region cluster.

To be able to rescue civilians, the agents should explore the map. Therefore, they keep a list of buildings to be explored. This list is first populated with the buildings of the assigned regions, but is later updated according to the messages received from the other agents. If the agents hear a civilian, they send messages to the other agents. The agent closer to the civilian adds the civilian's building to the exploring list.

Priority Based Communication In RSL, there are two types of communication; voice and radio. Both communication modalities has certain failure probability and limited bandwidth. Therefore, a communication strategy is needed to be able to efficiently communicate. We first create the list of message types such as, *Civilian Information*, *Building Burning*, and *Clear Path Is Needed*. We evaluate the available communication channels and assign message types to the channels. We also define the priority levels of the messages. For example, *Civilian Is Saved Or Dead* message has the highest priority. When an agent wants to send a message, the message is enqueued to the message queue. The message controller checks the message queue and sends messages depending on the available communication channels and bandwidth. To be able to increase the reliability of the message in the presence of noise, the message controller repeats the bytes of the message. The number of repeats is calculated as follows:

$$repeatcount = \lceil \frac{log(1 - reliability)}{log(noise)} \rceil$$

The Subsumption Behavior Component Subsumption architecture is a behavior arbitration mechanism. The most important property of subsumption architecture is that behaviors can be tested without interfering with each other. Therefore, we developed a repository of behaviors. Some of those behaviors are common to every agent type. For example, every agent should save themselves as the very basic behavior. Every agent should search the map when they do not have an active high priority behavior. Every behavior has a certain activation condition. For example, *Clear Blocked Road* behavior is activated when the agent type is police force, and is the agent close enough to clear the blockage.

We define three basic tasks which requires tight task allocation for better performance. These tasks are rescuing the civilians, extinguishing the fires, and clearing the blockages. Therefore, we use a market-based algorithm to effectively allocate these tasks to the agents. The auction behavior is a part of the subsumption architecture. If any behavior is not active for rescuing the agent's own well-being, the agent will execute its auction behavior for the task whose cost is minimal for the agent. Auction will be carried out by sending a message to the other agents. The other agents will calculate their own costs and either approve or give a bid smaller than the cost of the agent which started the auction. When the bid could not be improved or certain time step is completed, the auction ends, and the agents execute their assigned tasks. More details of market-based algorithm can be found in Section 4.

4 The Market-Driven Method

Market-driven methods (MDM) are based on the maximization of the overall gain of a group of robots by cooperation, collaboration and/or competition among them. This requires taking the total profit of that group into consideration while planning. Communication between robots for trading jobs, power and information is essential. Either distributed or centralized decision mechanisms may be employed based on the structures of the teams [4].

4.1 Application of MDM

In our implementation we integrate MDM and behavior based (BB) methods as shown in Fig. 1. An additional behavior that applies the market logic is integrated to the behavior based system. For every task this market implementation is specialized in order to meet the specific needs. The details of using MDM is given in [8].

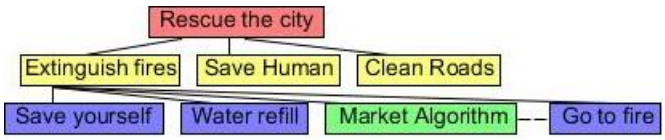

Fig. 1. The hybrid architecture

4.2 Generalized Market Algorithm and Auction Mechanism

The most important goal in a search and rescue mission is optimizing the process through high level planning. In the RSL competitions, the optimization measure is the score of a team at the end of the simulation. The whole system is a "heterogeneous multi-agent system" as there are different agents specialized in different tasks and there is more than one agent contributing to the same aim. Our goal is integrating market driven methods in order to balance reactive characteristics of the system with deliberative one [9].

Application of the Market Based Algorithm In the implemented market algorithm, every agent without an assignment calculates the costs for its known fires, and sends the best two of these costs to the center. The center, using its auction tools adds those bids to the appropriate auctions and gathers results for the ongoing auctions. If according to the results one agent is assigned to more than one building, an auction weighing the priority of the building and the cost for agent in taking action against that building is held on those results and the final decision is sent to the agent. If according to the results one agent is not assigned to any building, it is added in the auctions held for three buildings with the highest priority and no utilization. If there are results involving more than one agent, they are interpreted using the method described above.

As can be seen in the test results in [9], the market algorithm is a very important factor in enhancing the scores by establishing communication hence cooperation and collaboration between agents. It is this collaboration that improves the scores, as it avoids "excessive clustering" around disaster events and provides a close-to-optimal distribution of work, and resources around jobs in an intelligent manner, taking into consideration important factors like collective capacities of a groups versus jobs.

Due to the nature of the search and rescue task there are many parameters that need to be considered, however, the scores in the competitions show that this approach improves task achievement considerably.

5 Research Direction: The Layered Decentralized Partially Observable Markov Decision Processes Approach

We use the general approach for solving Dec-POMDP problems using genetic algorithms developed by Eker and Akın [5], [6]. We previously showed that we can learn multi-agent control policies using this approach for robot soccer [7]. We use the same approach to learn layered control policies.

In the initial implementation, we only consider the problem of dynamic task allocation of fire brigades to buildings. We use the map as a graph where every node is a building and if there is road between buildings these nodes are connected. There are actions as many as the number of buildings for every agent. Starting from the top layer, we divide the set of actions into two sets. The division is repeated at every layer until there is only one action left. At every layer we solve a Dec-POMDP problem, which chooses the action cluster for the agents. We use genetic algorithms to search in the newly constructed policy space layer by layer. We learn the policies of one layer while keeping the other layer's policies the same. The details of the algorithm are given in Figures 2 and 3.

Algorithm Layered Dec-POMDP Algorithm
  $Q \leftarrow \langle q_1, q_2, .., q_L \rangle$
  Initialize Q with random policies
  repeat
    for all $l \in [L, 1]$ **do**
      $\langle q_l, V \rangle \leftarrow$ SolveByGA(Q, l)
    end for
  until No improvement

Fig. 2. Layered Dec-POMDP algorithm

Algorithm SolveByGA
Input Q layered policies, l layer index
Output q_l^{best} improved policy, V reward of the policy
  Generate Quasi-random initial population p for q_l
  repeat
    Compute reward of p by simulation
    Evolve p
    $q_l^{best} \leftarrow$ get best solution
    $V \leftarrow$ compute reward of q_l^{best}
  until No improvement

Fig. 3. Dec-POMDP genetic algorithm

5.1 Layered Dec-POMDP Policy Representation

We represent the flat Dec-POMDP policy in layers where we solve different Dec-POMDP problems. This representation compresses the flat Dec-POMDP policy with clustering action and observation sets. We represent every layer as a finite state controller policy with the clustered Dec-POMDP model. As opposed to the standard approach where a different policy is used for every agent, we use one policy for every agent of the layer since there are many agents. At every layer, we compute a Dec-POMDP policy whose observation set is calculated by a function, *obsCluster*, and action set calculated by the function, *actionCluster*. The *obsCluster* function will generate a new observation set based on a subset of observations which are calculated according to the previous layer. In the same manner, *actionCluster* function will generate a new action set based on a subset of actions which are calculated according to the previous layer.

The approach can work with arbitrary *obsCluster* and *actionCluster* only if it satisfies the following conditions; *actionCluster* will divide the given action set into two disjoint sets and *obsCluster* will provide information to summarize the observation due to two disjoint action sets. We can see that we also hierarchically divide the state of the Dec-POMDP model according to clustering functions. Although our policy representation does not use the state set, it implicitly acts on the state set of the problem. Therefore, this hierarchical policy representation can represent optimal policies when the clustering functions are also generated as disjoint state sets at every layer.

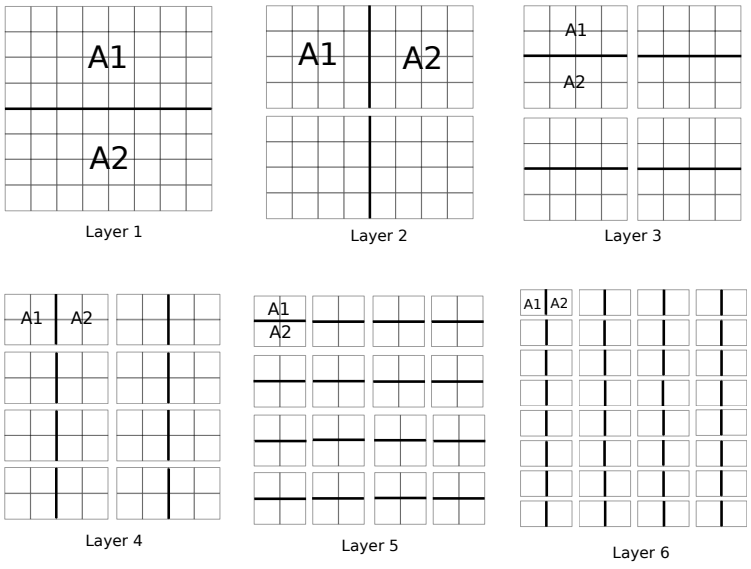

Fig. 4. Clustering actions for firefighting dynamic task allocation problem. Every grid represents a house.

We explain the concept of layered Dec-POMDP policy with an illustrative example as follows. We first determine our clustering functions. As we previously pointed out, these should be defined so that the disjoint action and observation sets are defined over the disjoint state set. We consider the firefighting problem, whose state space is defined on a grid. Every grid or house has a fire level and the agents have positions on the grid. Therefore, the state set is $S = \{h_{i,j} \times a_k : 0 \leq i, j < 8 \text{ and } 0 \leq k < |N|\}$ for a 8x8 problem. $h_{i,j}$ denotes the fire level of the houses, N denotes the agent set and a_k position of k^{th} agent. The observation set is the fire level of houses and the position of the agent. the action set is the houses. We define *actionCluster* so as to divide the house grids geometrically into two equal sets as seen in Figure 4. Since we divide into two sets at every layer, we have $2 \times log|numGrid|$ layers. At the last layer, the policy chooses the primitive action. At every layer, we solve a Dec-POMDP problem which is actually distributed to the different locations of the grid.

6 Current Scores of RoboAKUT 2015

We have tested the RoboAKUT 2015 code on the scenarios used in the finals of the RoboCup 2014 RSL competition. Table 1 demonstrates that the current state of RoboAKUT team is a bit far from the runners up of RoboCup 2014. The first four rows display the scores of the finalists in the official RoboCup 2014 results. The last row shows the scores of the RoboAKUT 2015 team. Figure 5 displays the map at the end of the simulation as a result of a run in Berlin3 scenario. When we investigate the results, we see that we mostly suffer from clearing blockages since we have very low scores from Joao3 and Istanbul3 scenarios. However, we have comparable scores for Kobe4, Eindhoven3, Paris3, and Mexico2 scenarios.

Table 1. RoboCup 2015 Final Scenario Results

	Maps							
	Kobe4	**Eindhoven3**	**Paris3**	**Berlin3**	**NY3**	**Joao3**	**Istanbul3**	**Mexico2**
S.O.S.	204.20	41.09	127.14	83.44	102.43	87.47	58.50	273.73
MRL	197.90	33.71	122.96	88.39	143.97	63.17	39.03	278.59
CSU-YUNLU	203.11	34.83	122.02	80.36	122.89	58.42	19.59	276.94
Kherad	181.74	29.56	122.21	95.03	137.08	52.24	44.27	270.72
RoboAKUT 2015	107.60	24.01	**118.80**	**83.42**	59.85	1.68	4.15	**220.23**

7 Conclusions

In this paper we presented an overview of the agent system model and the algorithms of the RoboAKUT 2015 team. The agents are market based and can utilize resources effectively even under dynamic conditions for fire fighting, saving civilians and clearing roads. The work on applying a layered Dec-POMDP based algorithm is in progress.

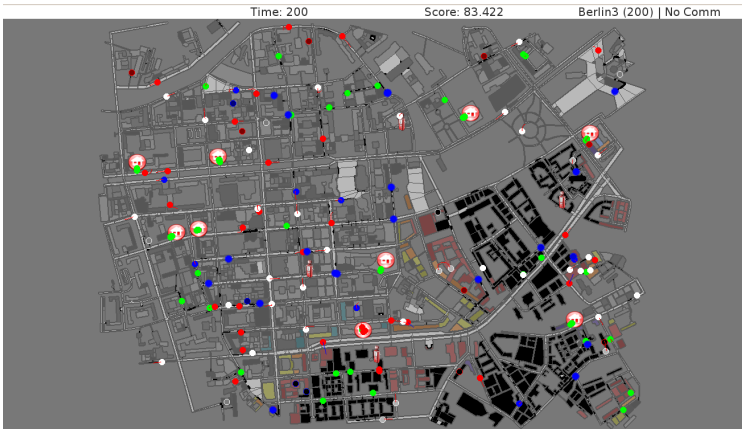

Fig. 5. Berlin 3 map result.

The test runs on the simulator show that code is more robust and the fires are successfully extinguished and the majority of the civilians are saved with an overall significant performance increase over RoboAKUT 2014.

References

1. H. Levent Akın, Orçun Yılmaz, and Mehmet Murat Sevim. Roboakut 2010 rescue simulation league agent team description. Technical report, Bogazici University, 2010.
2. H. Levent Akın and Okan Aşık. Roboakut 2014 rescue simulation league agent team description. Technical report, Bogazici University, 2014.
3. Rodney A Brooks. A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of*, 2(1):14–23, 1986.
4. Hatice Kose, Kemal Kaplan, Cetin Mericli, Utku Tatlidede, and Levent Akın. Market-driven multi-agent collaboration in robot soccer domain. In *Cutting Edge Robotics*, pages 407–416. pIV pro literatur Verlag, 2005.
5. Barış Eker and H. Levent Akın. Using evolution strategies to solve DEC-POMDP problems. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 14(1):35–47, 2010.
6. Barış Eker and H. Levent Akın. Solving decentralized pomdp problems using genetic algorithms. *Journal of Autonomous Agents and Multi-Agent Systems*, 2012.
7. O. Aşık and H.L. Akın. Solving multi-agent decision problems modeled as dec-pomdp: A robot soccer case study. In *RoboCup 2012 Symposium, June 24, 2012, Mexico City, 2012*, 2012.
8. H. Levent Akın and Mehmet Murat Sevim. Roboakut 2012 rescue simulation league agent team description. Technical report, Bogazici University, 2012.
9. Burak Zeydan and H. Levent Akın. Market-driven multi-agent collaboration for extinguishing fires in the robocup rescue simulation domain. In *2nd CSE Student Workshop (CSW'11)*, 2011.