RoboAKUT 2013 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 Bogazici University. In this paper, we give a description of the software architecture and the algorithms used by the RoboAKUT 2013 team. Our current rescue team is based on the market paradigm together with regional task management. For the RoboCup 2013 competition, based on our experiences in RoboCup 2012 we made 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. Our results with the RoboCup 2012 maps show that RoboAKUT 2013 team will be a serious contender in RoboCup 2013.

1 Introduction

RoboCup Rescue Simulation agent competition consists of a disaster management simulation with 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 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 Bogazici 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 2013 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 proposed Decentralized Partially Observable Markov Decision Process (Dec-POMDP) based approach is described in 5. The current scores of RoboAKUT 2013 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 2013

RoboAKUT team code was completely rewritten for the RoboCup 2010 competition [1]. RoboAKUT 2013 is the optimized and improved version of the RoboCup 2012 [2] code based on the competition experiences.

RoboAKUT 2013 currently uses a market driven approach 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 effectively is a hybrid market paradigm and regional task assignment system. We are now in the process of adapting a new approach based on Decentralized Partially Observable Markov Decision Process (Dec-POMDP) developed in [3, 4] and which was succesfully applied to robot soccer [5].

4 The Market-Driven Method used in RoboAKUT 2013

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 teams [6].

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 [2].

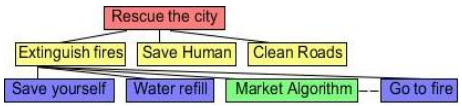

Fig. 1. The hybrid architecture

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 ones.

For this purpose, we first started with a general market based algorithm implementation. We determined the most common requirements for a market algorithm. Then we designed a general system containing these required objects in abstract forms. Our abstract system consists of four major components which form the basic requirements of any market algorithm implementation. These are:

- **Visualization component** provides us with the opportunity of observing the ongoing state of our simulation, i.e., the decisions made, the fires around the city, the placement of agents, and the actions by agents are visualized using this tool. The distinction between the states of objects is achieved through the use of colors and tones of colors. We use this tool for debugging purposes.
- **Auction component** used for holding auctions, taking bids from the agents, and coming up with results, taking all the data concerning that auction into consideration according to a user-defined function. This general auction mechanism can be modified for the specific needs of different auctioneers.
- **Cost function component** calculates the costs for alternative tasks given the percepts and past knowledge,. The tool can be considered as a black box that gets inputs and gives out a numeric cost as output. Like the auction tool, this is implemented as a general one that can be modified according to specific requirements concerning different bidders.
- **Communication component** is used for simulating and imitating communication in the real system for debugging purposes.

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.

During the cycles of central decision making, an agent starts its action for the building with the least cost to it and according to the final decision by the center, it either preempts its current action or not.This approach does not put much strain on the existing communication structure.

As can be seen in the test results in [7], 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 improvess task achievement considerably.

5 The Proposed Approach for Solving Problems Modeled as Decentralized Markov Decision Processes

The *Decentralized Partially Observable Markov Decision Process (DEC-POMDP)* [8] model consists of 7-tuple $(n, S, A, T, \Omega, Obs, R)$ where:

- n is the number of agents.
- S is a finite set of states.
- A is the set of joint actions which is the Cartesian product of A_i $(i = 1, 2..., n)$ i.e. the set of actions available to $agent_i$.
- T is the state transition function which determines the probabilities of the possible next states given the current state S and the current joint action a.
- Ω is the set of joint observations which is the Cartesian product of Ω_i $(i = 1, 2..., n)$ i.e. the set of observations available to $agent_i$. At any time step the agents receive a joint observation $o = (o_1, o_2, ..., o_n)$ from the environment.
- Obs is the observation function which specifies the probability of receiving the joint observation o given the current state S and the current joint action a.
- R is the immediate reward function specifying the reward taken by the multiagent team given the current state and the joint action.

5.1 Dec-POMDP Policies and Finite State Controllers

A Dec-POMDP policy is a mapping of the observation history to the actions. Policies are usually represented as a policy tree. Since the tree representation is not sufficiently compact, a Finite state controller (FSC) representation can also be used. A FSC is a special finite state machine consisting of a set of states and transitions. These abstract states called FSC nodes are different from the actual environment states. Every FSC node corresponds to a single action which is the best action for that particular state. Transitions are based on making a particular observation at a particular FSC node. An example finite state controller with only two observations and three actions is given in Figure 2. In a FSC, there is always an initial state. If we assume that $S1$ is the initial state then first $A1$ is executed. If the agent gets an observation $O2$, it updates its current FSC node to $S2$ and executes the action $A2$. Action execution and FSC node update continues until the the end of the episode. This finite state controller represents the policy of a single agent.

It is possible to model a Dec-POMDP policy using FSCs with different numbers of nodes. Since every node corresponds to one action, the minimum number of nodes is the number of actions. Since having greater number of nodes than the number of actions does not improve the performance of the algorithm[4], we use the number of FSC nodes as equal to the number of actions. This approach has been succesfully applied to robot soccer in [5].

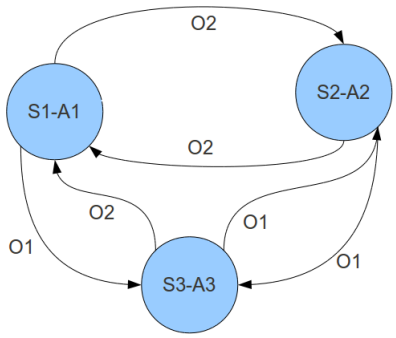

Fig. 2. An Example Finite State Controller

5.2 Genetic Algorithms

In a genetic algorithm, a candidate solution is encoded in a chromosome and the set of all chromosomes is called a *population*. The fitness of a candidate solution determines how good the candidate is. Through the application of evolutionary operators such as selection, crossover, and mutation, a new population is created from the current population. When the typically fitness based convergence criteria are met, the algorithm terminates and the best candidate becomes the solution of the algorithm [9].

Encoding In order to solve a Dec-POMDP using genetic algorithms, the candidate policy must be encoded. We encode a FSC as a chromosome as follows: the first n genes represent node-action mapping and their values are between 1 and the number of actions (A). The next genes contain for each node, the observation-node mapping which denotes the transition when an observation is made as seen in Figure 3. The value of this range is between 1 and S i.e. the number of nodes. The whole chromosome of the Dec-POMDP policy is constructed by concatenating the policies of each and every agent.

Fitness Calculation Fitness calculation is an essential componenet of any genetic algorithm. When the transition and reward functions of a Dec-POMDP problem is available, the fitness values for a given policy can be calculated. However, for problems with

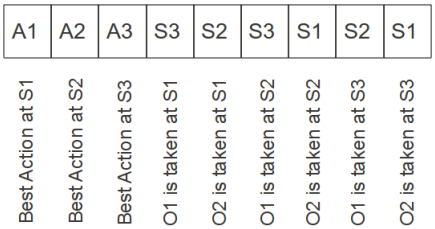

Fig. 3. An Example *FSC* Encoding

unknown transition and reward functions, only approximate fitness calculation is possible. One method of calculating fitness approximately is by running a large number of simulations with a given policy [4].

5.3 The GA-*FSC* Algorithm

The evolutionary strategy based approach proposed in [3] has been shown to be not sufficiently scalable as the number of agents increase. In [4] a better approach that uses FSCs is proposed. In our implementation we use this approach.

This algorithm has two major components :

- **Encoding the candidate policy:** A policy is represented as a *FSC* and is encoded as an integer chromosome.
- **Searching the policy space for the best policy with genetic algorithm:** In [4], two fitness calculation approaches are proposed: exact and approximate. For the rescue simulation problem considered here, exact calculation is not possible since the dynamics of the environment are not known exactly. The approximate calculation method, however, relies on running many simulations with a given policy and taking the average reward of those simulations as the fitness of the policy.

The algorithm has three stages: pre-evolution, during evolution, and post-evolution. After a random population is formed, the k best chromosomes are selected based on their fitnesses. Those k chromosomes are copied to the best chromosomes list. At the end of each generation, the best k chromosomes of the population are compared to the chromosomes in the best chromosomes list. If it is better than one of the current best chromosomes, its fitness is calculated more precisely by running additional simulations. If it is still better, it is added to the current best chromosomes list. At the end of the evolution which is determined by setting a maximum generation number, the best of best chromosomes list is determined by running additional simulations. In this study, we keep 10 chromosomes in the best chromosomes list.

5.4 Implementation of the GA-*FSC* Algorithm

The GA-*FSC* Algorithm will be used to determine the policies for the firefighter, police and ambulance agents separately in an off-line manner. The obtained FSCs will be used during the actual competition runs.

6 Current Scores of RoboAKUT 2013

Table 1. RoboCup 2012 Final Scenario Results

	Maps		
	Berlin5	Kobe3	VC3
AUT S.O.S.	56.395	89.433	4.645
Ri-one	36.499	47.35	6.385
MRL Agent	14.16	93.882	5.529
ZJUBase	14.08	90.499	5.246
RoboAKUT 2013	18.171	16.651	1.858

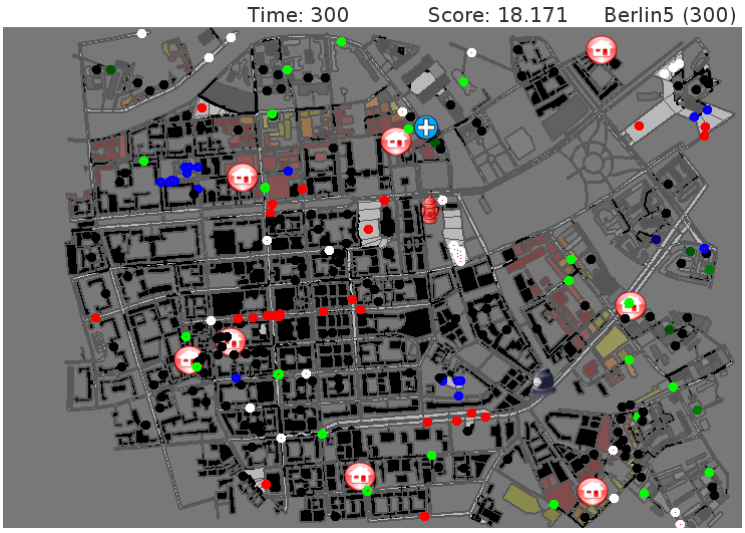

Fig. 4. The final state of Berlin 5 scenario with RoboAKUT agent

We have tested RoboAKUT 2013 code on the scenarios used in the finals of the RoboCup 2012 RSL competition. Table 1 demonstrates that the current state of RoboAKUT team is on par with the runners up of RoboCup 2012. The first four rows display the scores of the finalists in the official RoboCup 2012 results. The last row shows the scores of the RoboAKUT 2013 team. Figure 4 displays the map at the end of the simulation as a result of a run in Berlin 5 scenario.

7 Conclusions

In this paper we presented an overview of the agent system model and the algorithms of RoboAKUT 2013 team. They consists of market based agents which can utilize resources effectively even under dynamic conditions for fire fighting, saving civilians and clearing roads. The work on applying a DEC-POMDP based algorithm is in progress.

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 2012.

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 Mehmet Murat Sevim. Roboakut 2012 rescue simulation league agent team description. Technical report, Bogazici University, 2012.
3. 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.
4. Barış Eker and H. Levent Akın. Solving decentralized pomdp problems using genetic algorithms. *Journal of Autonomous Agents and Multi-Agent Systems*, 2012.
5. 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.
6. H. Kose, K. Kaplan, C. Mericli, U. Tatlidede, and L. Akin. Market-driven multi-agent collaboration in robot soccer domain. In *Cutting Edge Robotics*, pages 407–416. pIV pro literatur Verlag, 2005.
7. 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.
8. Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The Complexity of Decentralized Control of Markov Decision Processes. *Math. Oper. Res.*, 27:819–840, November 2002.
9. H. Levent Akın. Evolutionary Computation: A Natural Answer to Artificial Questions. In *Proceedings of ANNAL: Hints from Life to Artificial Intelligence*, pages 41–52, METU, Ankara, 1994.