Agent Coordination and Disaster Prediction in Persia 2007, A RoboCup Rescue Simulation Team based on Learning Automata

Moahammad R. Khojasteh, and Aida Kazimi

***Abstract*—Rescue teams must effectively cooperate despite sensing and communication limitations in order to save lives and minimize damage in RoboCup Rescue, which is a simulation of urban disasters. This paper describes the main features of the Persia 2007 rescue simulation team that was qualified to take part in RoboCup 2007 competitions in USA and ranked 10th at final competitions. We have explained Persia 2007 Agents' Architecture, their decision making behavior, and their prediction strategies. We describe the major challenges each of the agent types should be concerned about in the RoboCup Rescue Simulation domain. Then, we have proposed our solutions based on either algorithms or machine learning techniques for each challenge. Our machine learning techniques are mostly based on Learning Automata that have been used in some parts of our agents' development.**

***Index Terms*— Coordination, Disaster Prediction, Learning Automata, Multi-Agent Systems, RoboCup Rescue Simulation.**

I. Introduction

RoboCup Rescue Simulation is an excellent multi-agent domain which includes heterogeneous Agents that try to cooperate in order to minimize damage in a disaster. These Agents must have effective cooperative behavior despite incomplete information. In fact, the main goal in this domain is minimizing the damage by helping trapped agents, extinguishing fiery collapsed buildings, and rescuing damaged civilians.

To reach to this goal, agents must be flexible. It means that they should be able to work in different (even center-less) situations efficiently and make best decisions coordinated with each other. Decision making in such a domain is very complex, because each agent type has some limitations, e.g. agents' limited field of view, limited communication bandwidth, limited water quantity, and etc. We have tried to implement flexible agents despite these problems.

In a highly dynamic disaster situation as in RoboCup Rescue domain, it is a crucial capability of rescue teams to predict future states of the environment as precisely as possible. To do so, we have designed new prediction methods in our agents' long-term plans and short-term reactions. One of the most important predictions in such a domain is to predict how disasters evolve over time, e.g. how the fire spreads in the city. As a result of running different simulations and by using either machine learning methods (to determine fire spread rate) or their results' classification and data mining (to predict each injured civilian's dead-time), we have tried to create some predictions in this dynamic domain.

Persia 2007 is based on its previous versions [12][13] which has taken part in last two World RoboCup Competitions (2005 in Osaka-Japan and 2006 in Bremen-Germany). Some of the main ideas behind Persia 2007 team are investigating and evaluating machine learning methods based on Learning Automata [8] and Cellular Learning Automata [5][9] among others in cooperating agents in a team which could act in a complex multi-agent domain.

Learning automata act in a stochastic environment and are able to update their action probabilities considering the inputs from their environment, so optimizing their functionality as a result. And each Cellular Learning Automata is made of a cellular automata [10] in which each cell is equipped with one or more learning automata that identify the state of this cell.

Also, because of the large state space of a complex multi-agent domain, it is vital to have a method for environmental states' generalization. We have devised some ideas based on technique called "The Best corner in State Square" that we had used successfully before [3] in parts of some simulations for generalizing the vast number of states in agents' environment to a few number of states.

Some of these ideas had been used in other simulated multi-agent domains like RoboCup Soccer2D Simulation domain [1][2][4] and trade networks [5][9] prior to RoboCup Rescue Simulation domain. We have started to implement similar ideas in RoboCup Rescue Simulation domain from 2005 and have extended our use of Learning Automata-based machine learning techniques in much more parts of our simulated team comparing to what we had done before [12][13].

In this paper, we will talk a bit about our agents' multi-layered architecture that we propose to simplify their decision making process, their skills and action selection, their disaster prediction via learning, and their coordination. However, we've tried to concentrate on our learning approaches to our agents' major challenges more than the other parts.

II. Agent Skills and Action Selection

Agents in the rescue simulation environment should be able

Manuscript received February 4, 2010.

M. R. Khojasteh is with the Computer Engineering Division, College of Engineering, Islamic Azad University – Shiraz Branch, Shiraz, Iran (phone: +98-917-305-5001; e-mail: khojasteh@iaushiraz.ac.ir).

A. Kazimi is with Persian Nice Dreams Company, Shiraz, Iran (e-mail: akazimi@pndream.com).

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

to do a series of actions based on some limited information that they gain via vision or communication. In order to design and implement flexible agents such that they can act with high performance in different and hard (and even center-less) situations we have designed and implemented a layered architecture.

Our layered architecture consists of three layers including skills layer, event handler layer, and high level strategy layer and it can be shown as a whole in figure 1.

We have used the basic idea for layered learning in [6] and also we have added our own ideas to design this 3-layer architecture. We explain the role of each layer briefly.

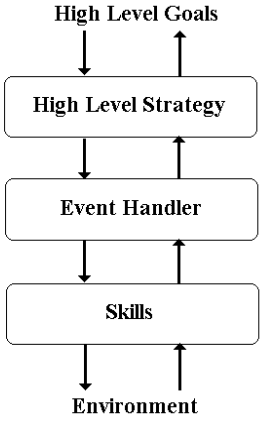

Fig. 1. Persia Agents' Architecture.

A. Skills Layer

In this layer we have defined some skills for each agent, based on its type (besides the skills that all agents have in common). This layer provides low level skills and facilities for higher level layers. Some of these skills are general like *moving, sensing, saying,* and *hearing*. But some of these skills have been specialized based on the agent's type, e.g. *extinguishing* for fire brigade agents, *rescuing* and *loading* for ambulance team agents, and *clearing* for police force agents.

B. Event Handler Layer

In this layer we have designed and implemented the more complex decisions that each agent makes regarding its current conditions, events, and situations. At first we consider all the situations that each agent can be in, and then we predict all the actions that each agent might do in those situations. Then we use a simple finite state machine (FSA) in order to determine the agent's action in each event.

It must be considered that this layer is still a low level decision maker layer comparing to the high level strategy layer and agents just choose (and perform) some simple actions (not complex ones as in the strategy level) in this level. Some examples of simple decision makings possible in this layer are *extinguishing a fire* when positioned well next to it, *going to refuge* when getting damaged or running out of water in tanks, *path finding* for which we have used a method based on Dijkstra shortest path algorithm that was first implemented by SOS team [7], and *searching* to find injured civilians.

Cooperation of agents in groups is accomplished through some high level decisions that are made in the next layer.

C. High Level Strategy Layer

In this layer, we have provided the more complex decisions (comparing to the above) that cause a typical agent to become a member of a group and work in it. In this layer we have tried to build a virtual environment in the agent's memory by considering the messages and the commands that are received from other agents, so that the agent could be able to know what has happened in the other points of the city.

It must be considered again that in this layer, our agents don't have to handle low level events in the environment. Also, they generally know what has happened in their environment. So, in this layer, each agent knows how many groups have been formed to work in the environment and what are they going to do (and of course where they are now and/or where they will be in the near future).

This high level information, simplifies the designing and implementing our machine learning methods (mostly based on Learning Automata).

In order to describe our approach for action selection, e.g. how our agents decide about the priority and selection of actions while facing the disaster, we have explained our Fire Brigades' approach as an example.

Fire brigades have an important role in preventing fire spread in the city and in rescuing civilians' lives. To extinguish fires, fire brigades should from groups to act more efficiently. In our approach, fire station forms such groups and assigns sufficient number of fire brigades to each fiery zone. Fiery zone is defined as a group of neighboring burned buildings. After these assignments, each group goes on his work and each fire brigade decides itself which fiery buildings in its allocated area must be extinguished first to prevent fire spread. This decision is based on building's danger. By danger we mean the potential damage a building might have if it catches fire. We experienced that building's danger formula in [12] should be substituted with new formula in some parts, as follows:

$$Vu = \frac{1}{1-\sqrt{\frac{1}{1+\log(1+V)}}} \quad (1)$$

Where Vu is the building's vulnerability [12] and V is the building's volume, which affects building's vulnerability.

$$S = \sum_{n \in neighbors} Vu_n \times (1+\log(1+I_n)) \quad (2)$$

Where I is building's infectivity [12].

On the other hand, the potential buildings for burning are the buildings which are neighbor to at least one of the border buildings of a fiery zone. The border of a fiery zone is defined to the set of all ignited buildings for which there is at least one unburned building in their neighborhood as depicted in figure 2.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Fig. 2. A fiery zone and its border buildings, which are marked with white points.

Actually, our fire brigades extinguish the boundary buildings, which are more dangerous in each fiery zone first in order to stop the spread of existent fire to other unburned buildings. This way, they try to achieve a higher final performance.

Moreover, some other parameters such as the distance between a fire brigade and a candidate target (i.e. a fiery building), the number of civilians which are threatened by the fire, and at last the candidate target reach-ability, have direct effects on fire brigades' decisions in order to select their final target.

III. Disaster prediction

The most important disaster prediction that we have implemented in our team is fire spread rate prediction by using Cellular Learning Automata. We have also predicted the injured civilians' dead-times with our data-mining techniques.

One of the well-known Learning Automata algorithms that we have used extensively is Lrp [1][2][3][4][14]. It can be explained by the following code fragment when a Learning Automata wants to update its action probabilities based on the response it has received from its environment and regarding the Lrp learning algorithm [8][13][14]. In writing this fragment we have assumed that we deal with 8 discrete states, we use one Lrp in each of these states, and we have 8 possible actions for each Lrp in any instant. Also, r (the response) = 0 when the automata receives a penalty and r = 1 when the automata receives a reward. The factor 0.1 in this code fragment shows the amount of increase/decrease for these 8 action probabilities too.

```
for ( int i = 0; i < 8; i++ ) {
   if ( i != myLastAction ) {
      probability[myStateInLastAction][i]
      += (1 - r) * (0.1) * ((1.0 / 7.0) -
      probability[myStateInLastAction][i])
      - r * ( 0.1 ) *
       (probability[myStateInLastAction][i]);
   }
}
  probability[myStateInLastAction][myLastActi
  on] -=
  (1 - r) * (0.1) *
  probability[myStateInLastAction][myLastActi
  on] + r * (0.1) * (1 -
  probability[myStateInLastAction][myLastActi
  on]);
```

We explain our prediction methods in the next sections.

A. Fire Spread Rate Prediction

After earthquake, some buildings catch fire in some different parts of the city. Naturally, fire will spread in the city in all directions and threat civilians' life if not controlled well. So, predicting fire spread rate is one of the most useful and at the same time difficult prediction in RoboCup Rescue Simulation System. We had proposed an approach for predicting fire spread rate, based on CLA method, for the first time in [12] although it had some defects. In Persia 2007 we have tried to refine it and make it more a practical and efficient approach.

It's obvious that we need to have some distinct states in our CLA method as in any other machine learning method. So, we have selected some parameters that seem to be important in order to define our states. These parameters consist of:

- $Volume_{Bi}$, it represents Bi's volume (Bi is one of the environment's buildings).
- F, it represents the average fieriness of Bi's neighbors
- T, it represents the total temperature of Bi's neighbors
- V_{bn}, it represents the average volume of Bi's burning neighbors
- V_{total}, it represents the average volume of all Bi's neighbors
- I_{bn}, it represents the average infectivity [12] of Bi's burning neighbors.

Then, we have defined a relationship between the above mentioned parameters as follows:

$$\frac{(F \times I_{bn} \times (\frac{V_{bn}}{V_{total}})) + T}{Volume_{B_i}} \quad (3)$$

We have divided the range between the experimental minimum and the experimental maximum values of the function above to 10 equal regions. Then we assigned each region to one state. So, at any instant in the simulation and by calculating the above function, we are put in a unique state.

For each state, we equip an Lrp Learning Automata [1][2][3][4][14] with 2 actions. By running a lot of different simulations and after convergence, our Lrp learns the probability of a building's being burnt depending on the above parameters (which include the important parameters in that building itself and also in its neighbors).

After going to the next cycle, the mentioned Lrp can give itself reward (if it had predicted correctly) or penalty (if it had predicted incorrectly).

B. Injured civilians' dead-times Prediction

We are trying to estimate each injured civilian's dead-time by using some data-mining methods with the factors HP, Damage, and Buried-ness as its inputs.

IV. Coordination among agents

We have categorized agents' coordination based on agents' types.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

A. Fire Station and Fire Brigades

We have categorized fire brigades' coordination based on being centralized or decentralized.

The Centralized Approach. In our approach, the Fire Station tries to assign an efficient number of fire brigades to each fiery zone. Unfortunately, the number and the position of fiery zones change in every simulation and also we do need several cycles to detect all the fiery zones.

First, our Fire Station should know the minimum necessity (i.e. minimum number of fire brigades) to assign to each fiery zone. We plan to use another Lrp Learning Automata in order to find this. In this Lrp, we can generalize its states for each fiery zone using the following parameters:

- Proportion of the average volume of fiery zone border to the average volume of the whole fiery zone
- The average danger of fiery zone's unburned buildings
- Proportion of the average fieriness of fiery zone border to the average fieriness of the whole fiery zone

Again, we divide the range between the experimental minimum and the experimental maximum values to 10 equal regions. Then we assign each region to one state. For each state, we should equip a Lrp with some actions. We let our learning actions be assigning the minimum number of fire brigades with average water (i.e. half of the maximum amount of water one fire brigade might have).

As a scenario for learning, we start some offline simulations with some different fiery zones (but just one fiery zone in each simulation). The center identifies the state of this fiery zone by mentioned parameters. Then, it looks to the corresponding Learning Automata in that state and assigns some (in the beginning of the simulation, it could be one or some predetermined number of) fire brigades to that fiery zone. If they could extinguish and control the fire by the end of simulation time, the station gives itself reward. But if not, it will give itself penalty, and so the probability that one more fire brigade will be assigned to the same fiery zone's state (by the station), is increased.

Doing so and after a lot of simulations, our Learning Automata would be able to assign the minimum number of fire brigades (with average water) in each state that a typical fiery zone is in.

Second, our Fire Station uses machine learning, but not like the method we used last year in [12] in order to find the best formation to assign a group to each fiery zone. Last year we tried to minimize all possible actions to just 3 types of actions. In the first type, all agents will be assigned to only one fiery zone at one time. The second type, assigns agents as distributed groups so that every detected fiery zone will have at least one fire brigade assigned. And the last type, assigns agents in groups, but there might be some detected fiery zones without any fire brigades allocated to them. From our simulations we experienced that the second type can not be efficient. Therefore in this year's approach, we have just 2 action types and each type consists of 4 different actions as follows:

1. Assign agents from the highest priority (zones) to the lowest priority (zones) in a preemptive manner. By preemptive, we mean the agents will leave their zone, if another preferable fiery zone has been detected at any instant.
2. Assign agents from the highest priority (zones) to the lowest priority (zones) that is calculated with their priorities calculated above in a non-preemptive manner. By non-preemptive, we mean when agents are assigned to one fiery zone, they won't leave there till they have finished their job at that assigned fiery zone (extinguishing the fire completely).
3. Assign agents from the lowest priority (zones) to the highest priority (zones) in a preemptive manner.
4. Assign agents from the lowest priority (zones) to the highest priority (zones) in a non-preemptive manner.
5. Divide agents from the highest fiery zone's minimum necessity (i.e. the minimum number of fire brigades necessary) to lowest fiery zone's minimum necessity and assign them in a preemptive manner.
6. Divide agents from the highest fiery zone's minimum necessity to lowest fiery zone's minimum necessity and assign them in a non-preemptive manner.
7. Divide agents from the lowest fiery zone's minimum necessity to highest fiery zone's minimum necessity and assign them in a preemptive manner.
8. Divide agents from the lowest fiery zone's minimum necessity to highest fiery zone's minimum necessity and assign them in a non-preemptive manner.

Consequently, Fire Station will find the best formation for each situation (state) after many simulations and assigns fire brigade agents. After these assignments each group goes on its work and each fire brigade decides itself which fiery buildings in its allocated area must be extinguished first to prevent fire spread.

The Decentralized Approach. In center-less situations or if our Fire Station crashed, one of our fire brigades will be chosen as a leader and does the Fire Station's task.

B. Ambulance Center and Ambulance Teams

We have categorized ambulance teams' coordination based on being centralized or decentralized.

The Centralized Approach. Ambulance Teams are responsible for rescuing and saving the lives of the victims that are buried under the debris caused by the collapse of buildings. Their way of operation directly affects the number of civilians in the city whose lives can be saved after the happening of the disaster. No doubt this is the most important goal of a rescue team. It seems that Ambulance Team agents must work in some formed groups in order to be more effective instead of working altogether. But our previous experiences show that in early cycles of the simulation, working individually might be more efficient.

Our Ambulance Team's strategy is based on its previous version [12] with some new changes as the results of our previous simulations' analysis. Like the previous approach in [12], we have divided the whole simulation time into two major phases for our Ambulance Team agents. But the way our agents act in each phase is somewhat changed.

In the first phase of the simulation (i.e. in its early cycles), Ambulance Teams have limited moving capability because of the relatively high number of the blocked roads in the city. Our Ambulance Teams worked together in this phase last year [12], but this year we have concluded that the Ambulance Center must prepare (and update) a list of the best targets (i.e. injured civilian) that are recognized by now to each one of the

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Ambulance Teams. Each one of the ambulance agents will then look into its list to find the first reachable target considering its location.

The second phase of the simulation will start as soon as more blockades have been cleared by Police Forces. After a lot of random simulations and analyzing their results, we finally decided that Ambulance Teams should do their rescue job in some groups that have been formed based upon the number of required ambulances for each injured civilian. This is actually the last option, which we proposed previously in [12]. In order to calculate the number of required ambulances for rescuing a target, the Ambulance Center calculates buried-ness and dead-time for each injured civilian as in [12] and then assigns Ambulance Teams to the targets with the highest priorities. In this way, the Ambulance Center tries to coordinate Ambulance Teams in order to reach the best possible result.

It is known that selecting the best target is the most important task for Ambulance Team agents in RoboCup Rescue Simulation and maybe the most important targets to rescue are trapped agents. When there are no other trapped agents left, now it's the time for the Ambulance Center to prepare separate lists of the best targets (i.e. injured civilian) that are recognized by now for each one of the Ambulance Teams regarding their position. We have used a simple approach using a greedy algorithm to choose the best target by estimating civilians' dead-times for this case. It is necessary to say that by an injured civilian's dead-time we mean the remaining time left to the time of his death in case of no rescue operation.

The Decentralized Approach. In center-less situations our Ambulance Teams should select a leader with a simple voting mechanism. This leader then goes either to the Fire Station or to the Police Office and tries to communicate with them by saying and hearing the information. In fact we have replaced the Ambulance Center with this leader who assigns the Ambulance Teams to their targets and leads them during the simulation in a way similar to what the center did before.

C. Police Office and Police Forces

We have categorized police forces' coordination based on being centralized or decentralized.

The Centralized Approach. Last year we used some strategies based on Learning Automata [1][2][3][4] by police forces to clear the blocked roads. Our agents tried to learn how to find the most important blockade roads, and how to help other agents like Ambulance Teams or Fire Brigades.

By using our methods [12], police agents can find the most important blockades and clear them. But as the result of the analysis we had after the competition, we found out that in some occasions our Police Forces try to clear some roads that are not so important or strategic (e.g. a road that doesn't lead to any civilian, fiery building, refuge or trapped agent).

Consider a situation in which a Police Force agent opens a road that leads to the position of a civilian. Meanwhile an Ambulance Team agent, far from the mentioned civilian, is going to move towards him. There should obviously be some open roads leading to him at this instant, but due to incomplete information about blockades and open roads, the Ambulance Team agent does not know which path is open and has to test the roads that result from its path finding module one by one. This try and error method to find an open path towards the civilian causes that the Ambulance Team agent loses some cycles in this phase and so these agents lose their performance. These situations mostly happen in the first cycles of the simulation.

So this year we decide to reduce the time which agents lose in the first cycles of the simulation due to some problems like the one mentioned above. To solve this problem we have decided to find the more important paths (called highways) that act as bridges between the most important parts of the city (called regions). To find these highways we use algorithms as in [11].

It's necessary to mention that our overall approach is much like our last year's as in [12] in center-based situations. In center-less situations, we have proposed a new variation of our last year's center-based approach in [12] that we will describe in the following section.

The overall scenario can be explained like this: at the beginning of the simulation we assign some number of the Police Force agents to clear the highways. So in the first cycles of simulation whenever each agent wants to travel from one region to another, it at first tries to reach itself to the nearest highway. Then it moves through the highways to reach to the region which its target is located in. Inside the region, agents use normal path planning to move towards their target.

The Decentralized Approach. In center-less situations, we should provide a different approach. In our center- based approach [12], each police agent needs to be aware of other police force positions in order to do an action. In that approach police center transfers each police force's position to the other police forces, so after a very short time police forces can receive the information they need.

In the center-less situations police agents can't send their position to each other in each cycle because of the communication limitations, so it seems that the previous approach does not work efficiently in center-less situations. Because of this problem we have designed and implemented another approach for center-less situations.

In this approach a police agent will act as a commander. This commander assigns jobs to police agents.

At first, the entire city is divided in to four major parts and each of these major parts is subdivided into 20m*20m cells as shown in figure 3.

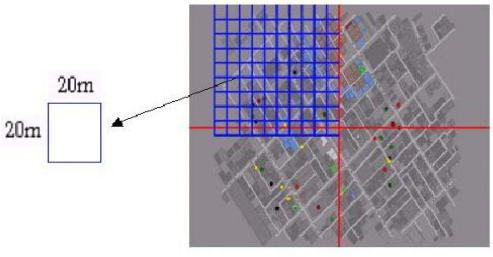

Fig. 3. A typical city divided virtually into 20(m)*20(m) cells.

We have chosen 20m*20m cells because agents are able to see everything in the radius of 10 meters from themselves. So, almost the whole square should be visible for an agent that is in it. For each police force, we assume a virtual attraction vector pulls it towards the position of each of the following items:

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

- Fiery buildings
- Refuges
- Trapped agents
- Fire brigades' and ambulances' important requests

The magnitude of each vector is conversely proportional to the distance between the above mentioned items and the police force, multiplied by a constant coefficient.

Coefficients are different based on the type of each item. These coefficients were chosen just by experimental data mining in this center-less approach.

Because of process time restrictions, we have considered no attraction vectors towards blocked roads. Instead, we have considered attraction vectors towards the centers of each cell (from the police forces) in addition to the above mentioned vectors. The magnitude of this vector is conversely proportional to the distance between the police force's position and the center of each cell. Also the coefficient for this magnitude is directly proportional to the number of roads inside the cell. It's obvious that after assigning a police agent to a cell, we will eliminate its corresponding vector. Also, if a path toward a refuge or fiery building was opened, its corresponding vector will be ignored thereafter.

The commander calculates the resultant vector for each police agent. Naturally the resultant vector points to one of the four major parts of the city. We call this part as the active part. Then the commander assigns a police force to one of the cells of this active part via an Lrp [1][2][3][4][14] Learning Automata.

Thereafter, police agent begins to clear the roads of the allocated cell. Since the commander knows the police forces' work areas, it is able to calculate the resultant vector of each police agent easily. The commander considers the center of the cell which the police force works in as the police's position. Then calculates the resultant vector with the procedure above and assigns this police force to another cell.

This way, the commander determines polices' next jobs. After this process (determining polices' next jobs) is done by the commander, the new job is sent for police agents. The commander also gives such information to every police force in each cycle. On the other hand, police forces save the commands which were sent form the commander each cycle. Each police force extracts the latest commander command after finishing its job in its allocated cell and then it begins to accomplish it. Each police force informs its position (its cell) to the commander during opening the roads.

We have again used a Learning Automata to determine the next cell that the police force should clear. The commander should learn which one of the cells is most important to clear. The rewards and penalties for the commander are given regarding the allocated cell. As an example, if there is a trapped agent in one of the cells but the police agent was assigned to a cell which doesn't have any special item in (like trapped agents or fiery buildings), the commander will get penalty.

REFERENCES

[1] M.R. Khojasteh, and M.R. Meybodi, "Evaluating Learning Automata as a Model for Cooperation in Complex Multi-Agent Domains," in Gerhard Lakemeyer, Elizabeth Sklar, Domenico Sorenti, and Tomoichi Takahashi (eds.): *RoboCup-2006: Robot Soccer World Cup X*, Springer-Verlag, Berlin, 2007, pp. 410-417.

[2] M.R. Khojasteh, and M.R. Meybodi, "Using Learning Automata in Cooperation among Agents in a Team," in *Proceedings of the 12th Portuguese Conference on Artificial Intelligence, IEEE Conference Publication Program with ISBN 0-7803-9365-1 and IEEE Catalog Number 05EX1157*, University of Beira Interior, Covilhã, Portugal, 2005, pp. 306-312.

[3] M.R. Khojasteh, and M.R. Meybodi, "The technique "Best Corner in State Square" for generalization of environmental states in a cooperative multi-agent domain," in *Proceedings of Eighth International Annual Computer Conference of Computer Society of Iran (CSICC'2003)*, Mashhad, Iran, 2003, pp. 446-455.

[4] M.R. Khojasteh, "Cooperation in multi-agent systems using Learning Automata," M.Sc. thesis, Department of Computer Engineering and Information Technology, Amirkabir University of Technology, 2002.

[5] M.R. Khojasteh, and M.R. Meybodi, "Cellular Learning Automata as a Model for Trade Networks in a Space of Agents who Learn by Doing," in *Proceedings of Sixth International Annual Annual Computer Conference of Computer Society of Iran (CSICC'2001)*, Isfahan, Iran, 2001, pp. 284-295.

[6] P. Stone, "Layered Learning in Multi_Agent Systems," PhD. Thesis, School of Computer Science, Carnegie Mellon University, 1998.

[7] S.A. Amraii, B. Behsaz, M. Izadi, H. Janzadeh, F. Molazem, A. Rahimi, M.T. Ghinani, and H. Vosoughpour, "S.O.S. 2004: An attempt towards a multi-agent rescue team," Team Description Paper, 2004.

[8] K.S. Narendra, and M.A.L. Thathachar, *Learning Automata: An Introduction*, Prentice Hall, Inc., 1989.

[9] M. Taherkhani, "Proposing and Studying of Cellular Learning Automata as a Tool for Modeling Systems," M.Sc. thesis. Department of Computer Engineering and Information Technology, Amirkabir University of Technology, 2000.

[10] S. Wolfrom, *Theory and Application of Cellular Automata*, Singapore: World Scientific Publishing Co., Pte. Ltd., 1986.

[11] S.B.M Post, and M.L. Fassaert, "A communication and coordination model for 'RoboCupRescue' agents," M.Sc. thesis, Department of Computer Science, University of Amsterdam, 2004.

[12] A. Kazimi, and M.R. Khojasteh, "Persia 2006, Towards a Full Learning Automata-Based Cooperative Team," in *Proceedings of the First Regional Conference of Computer Science and Engineering (RCCSE'2009)*, Shiraz, Iran, 2009, pp. 87-99.

[13] M.R. Khojasteh, H. Heidari, and F. Faraji, "Persia 2005 Team Description," Team Description Paper, 2005, unpublished.

[14] M.A.L. Thathachar, and P. Sastry, "A New Approach to the Design of Reinforcement Schemes for Learning Automata," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-15, No. 1, 1985.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)