Virtual and Intelligent Traffic signs In Rescue Simulation System: Imitation of Human Society in Agent Society

Mostafa Asghari #1, Behrooz Masoumi#2, Mohammad Reza Meybodi #3

#1 *Azad University of Miyandoab, Iran*

#2 *Azad University of Qazvin, Iran*

#3 *Amirkabir University of Technology, Department of Computer Engineering and IT, Tehran, Iran*

m_asghary86@yahoo.com, bmasoumi@Qazviniau.ir, mmeybodi@aut.ac.ir

***Abstract*—Robocup Rescue Simulation System is a suitable test-bed for test and evaluation of multi-agent system's related ideas and techniques. Hence, the world robocup competitions are held each year and the used ideas and techniques are evaluated in the form of different teams. In rescue simulation system, at the start of simulation, police agents should search and explore the earthquake area and open the blocked roads. In this paper, which is the first application of distributed learning automata in rescue simulation system, the agents imitate the ways which real people use for solving traffic problems in their societies. Like the real peoples' solutions for traffic problem, the agents use virtual police and street signs in their virtual environment junctions. The results indicate the fact that the proposed method gives more exploration power to rescue agents.**

***Keywords*— search and exploration, Distributed learning automata, rescue simulation**

I. INTRODUCTION

The RoboCup Rescue simulation environment is a challenging multi agent domain where task need to be done collaboratively by heterogeneous agents [1]. For robotics and multi-agent researchers, RoboCup Rescue works as a standard platform that enables easy comparison of research results. The problem introduced by RoboCup Rescue brings up several research challenges that go from Intelligent Robotics to Multi-Agent Systems (MAS) research. These research challenges include real-time flexible planning, multi-agent coordination and team formation, path planning and navigation, heterogeneous resource allocation and machine learning at the team level. In fact, the main goal in this domain is minimizing the damage by helping trapped agents, extinguishing fiery collapsed buildings, and rescuing damaged civilians. One of the most important and basic factors leading to reach this aim is to search and open the blocked roads so that the ambulances and fire brigades be able to rescue the injured civilians and to extinguish the burning buildings. This is the duty of police agents. With the start of simulation, the police forces should explore the earthquake area and open the blocked roads. The most of participated teams in the world robocup competitions use the shortest path algorithms (like dijkstra algorithm) for path planning [2] [3]. This means that the places visited by the agents are restricted to the cases that are located between the source and target in the shortest path. In [4] proposed a new hybrid algorithm, which uses ant colonies and learning automata that can be used to give more flexibility and exploration power to the agents.

In order to enhance the efficiency of police agent in search and exploration of the environment, we have applied the imitation of traffic management method in human society to the rescue simulation environment. Furthermore, we have placed virtual and intelligent traffic signs in junctions (nodes) so as to help the agents to reach their destinations. We have done this by using the distributed learning automata.

In this paper, first in section II we have introduced the learning automata and distributed learning automata briefly, then we present the proposed method in section 3. In section 4 the proposed algorithm is evaluated and finally, The Conclusion is stated in section 5.

II. LEARNING AUTOMATA

Learning Automata are adaptive decision-making devices operating on unknown random environments[5]. The Learning Automaton has a finite set of actions and each action has a certain probability (unknown for the automaton) of getting rewarded by the environment of the automaton. The aim is to learn to choose the optimal action (i.e. the action with the highest probability of being rewarded) through repeated interaction on the system. If the learning algorithm is chosen properly, then the iterative process of interacting on the environment can be made to result in selection of the optimal action.

Figure 1 illustrates how a stochastic automaton works in feedback connection with a random environment. Learning Automata can be classified into two main families: fixed structure learning automata and variable structure learning automata (VSLA) [6]. In the following, the variable structure learning automata is described. In a variable-structure stochastic automaton, the probabilities of the various actions are updated on the basis of the information the environment provides. Action probabilities are updated at every stage using a reinforcement scheme. It is defined by a quadruple $\{\alpha, \beta, P, T\}$ for which α is the action or

978-1-4244-4809-8/09/$25.00 ©2009 IEEE

output set $\{\alpha_1,\alpha_2,...,\alpha_r\}$ of the automaton, β is a random variable in the interval [0, 1], p is the action probability vector of the automaton or agent, and T denotes an update scheme.

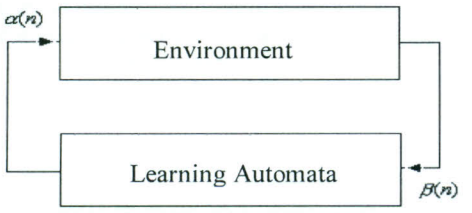

Figure 1. Learning automata - environment

The output of the automaton is actually the input to the environment. The input β of the automaton is the output of the environment, which is modelled through penalty probabilities c_i with $ci=[\beta\,|\,\alpha_i],i:1..r$. Important examples of linear update schemes are linear reward–penalty, linear reward–inaction, and linear reward–ε penalty. The philosophy of those schemes is essentially to increase the probability of an action when it results in a success and to decrease it when the response is failure .We used the following reward–penalty scheme for our algorithm :

-In desired (success) case:

$$p_i(n+1)=p_i(n)+a[1-p_i(n)] \quad (1)$$

$$p_j(n+1)=(1-a)p_j(n) \quad (2)$$

-In undesired (failure) case:

$$p_i(n+1)=(1-b)p_i(n) \quad (3)$$

$$p_j(n+1)=(b/r-1)+(1-b)p_j(n)\ \ \forall j\ j\neq i \quad (4)$$

The constants a and b are the reward and penalty parameters, respectively. Parameter r is the number of automaton's actions . When $a = b$, the algorithm is referred to as linear reward–penalty (L_{RP}) ; when $b = 0$, it is referred to as reward– inaction(L_{RI}) ; and when b is small compared to a , it is called linear reward–ε-penalty ($L_{R\varepsilon P}$). An L_{RP} scheme tends to equalize the penalty rates of the various actions, while an $L_{R\varepsilon P}$ scheme tends to equalize the penalty probabilities. Detailed information about learning automata can be found in [6].

A. Distributed LEARNING AUTOMATA (DLA)

DLA is a network of automata which collectively cooperate to solve a particular problem[7]. In DLA, the number of actions for any automaton in the network is equal to the number of outgoing edge from that automaton. When an automaton selects one of its actions, another automaton on the other end of edge corresponding to the selected action will be activated. At any time only one automaton in the network will be active.

Formally, a DLA with n learning automata(LA) can be defined by a graph(A,E), where A = { $A_1,A_2,\ldots,A_n$} is the set of automata and $E\subset A\times A$ is the set of edges in the graph in which an edge (i,j) corresponds to action a_j of automaton A_i.

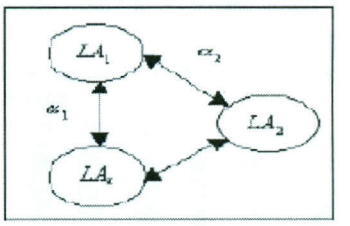

Figure2. DLA with there learning automaton

III. PROPOSED ALGORITHM

In the real world, there are police forces or real traffic signs in junctions in order to help the passengers to select paths toward their destination. Like real world, in rescue simulation environment, learning automatons are distributed over the simulated disaster-afflicted area. There is a learning automaton in each node of graph. The number of actions for each automaton is the same as the number of outgoing edges from node in which the automaton is placed on it.

When an agent wants to pass a node, the corresponding learning automaton is activated, then selects one of its actions and proposes the outgoing edge which corresponds to its selected action to the agent. Probability value for each road is computed on the basis of the road length and probability rule. Learning automaton deactivates its corresponding action to the edge which is used by the agent as incoming path to the current node. As a result, that road can not be selected again by the agent. After selecting one road by the agent in the current node, the active learning automaton is deactivated and the learning automaton which is placed at the other end of selected road is activated. This is repeated until the agent reaches its destination.

The important point is that, when a learning automaton is activated, it gives a penalty to its selected action, according to (3) and gives reward to the actions which are not selected, according to (4). Consequently, the selected road will have lower chance to be selected in the future and other outgoing roads from that node, which are not selected, will have more chance to be selected. Giving reward or penalty to roads will cause all of the outgoing roads have chances to be selected by the agents as a part of their path. Since the probability value for each road is based on its length, the short roads will have the high probability value; therefore, at the first time which the agent passes the node, the first short path and at the second time, the second short path will have the more chance to be selected. The purpose of this algorithm is the fast search and exploration of environment. Giving the penalty to the used roads, decreases the probability value for those roads and giving reward to unused roads increases their probability values. So each agent will try to pass through the roads that they

are not passed yet. Selection of next node (V_{i+1}) for each V_i is as follows:

1. Initialize the LA parameters (a=b=0.3).
2. Initialize the probability vector for all automatons actions on the basis of reverse of corresponding edges length.
3. Select one of outgoing roads from V_i randomly.
4. For all adjacent Edges(E_j) to V_i do
   If E_j has been selected as path
      Give penalty to E_j according to Eq.3
   Else
      Give reward to E_j according to Eq.4

Figure3. Psudo code for proposed algorithm

B. *Setting parameters for learning automatons*

As mentioned in the previous sections, the learning scheme which is used in this algorithm is linear reward-penalty (L_{RI}). Parameters a, b have the same value and are computed on the basis of try and error method. The result of simulation with different values for learning automata parameters (a , b) are presented in table 1.

TABLE1. DIFFERENT VALUES FOR LA PARAMETERS AND NUMBER OF NODES VISITED BY THE AGENT

time	a=b=0.8	a=b=0.7	a=b=0.6	a=b=0.5	a=b=0.4	✓ a=b=0.3	a=b=0.2	a=b=0.1
0 - 15	174	155	139	167	75	178	125	139
15 - 30	8	33	50	25	46	47	40	8
30 - 45	15	57	85	51	24	5	41	63
45 - 60	35	11	15	6	65	18	9	6
60 - 100	4	0	1	5	60	29	3	42
100 - 150	42	0	0	14	6	15	0	22
150 - 200	0	35	0	22	0	0	19	12
200 - 250	3	0	1	0	15	15	26	0
250 - 300	0	0	1	0	0	0	5	0

Table1 shows that the appropriate value for LA parameters is 0.3.

C. *Evaluation*

To evaluate the proposed algorithm, two different maps are used: Random-Small (China2008 Robocup competitions) and Kobe (default map for rescue simulation system) with 293 nodes and 381 edges in its simulated environment. The Persia agents which use shortest path algorithm to reach the destination (dijkstra) are used as benchmark to evaluate our proposed algorithm [2]. The target selection method for agents in our algorithm and benchmark algorithm (Persia) in all situations is the same.

For comparison, table 2 is about the exploration performance of Persia agents' algorithm during the simulation with two different mentioned maps. Exploration rate for Persia police agents with two maps is as figure 4 and 5.

TABLE2. NUMBER OF NODES VISITED BY PERSIA POLICE AGENTS DURING THE SIMULATION WITH TWO MAP

time / map	0 - 15	15 - 30	30 - 45	45 - 60	60 - 100	100- 150	150- 200	200- 250	250- 300
Kobe	31	22	31	11	38	54	47	35	23
Random	96	45	24	56	39	0	100	40	61

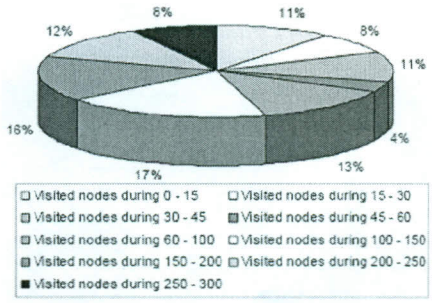

Figure4. Exploration rate for Persia police agents in Kobe map

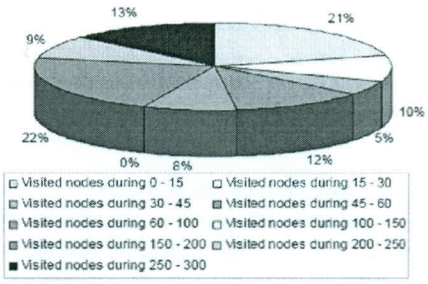

Figure5. Exploration rate for Persia police agents in Random-Small map

In the proposed method, the next node selection is done on the basis of probability which varies during simulation. Since each selected edge takes penalty and other edges, which are not selected, take reward, non selected edges will have more chance to be selected in the future. Also giving the penalties to the selected edges decreases the number of loops in path. The performance function of this algorithm during the simulation is as follows:

TABLE3. NUMBER OF NODES VISITED BY AGENTS WHICH USE OUR ALGORITHM DURING THE SIMULATION WITH TWO MAP

Time / map	0 - 15	15 - 30	30 - 45	45 - 60	60 - 100	100 -150	150 -200	200 - 250	250 - 300
Kobe	60	33	36	5	42	46	59	1	0
Random	141	60	75	40	40	10	81	2	11

Exploration rate for our agents with two maps is as figure 6 and 7.

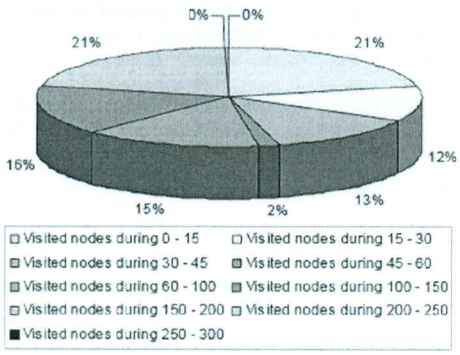

Figure6. Exploration rate for our agents in Kobe map

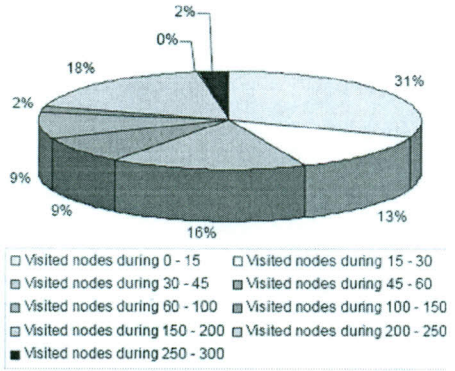

Figure7. Exploration rate for our agents in Random-Small map

In the previous section, Persia teams' routing algorithm and our proposed algorithm are simulated and evaluated separately. To show the efficiency of our algorithm from exploration view, we compared it with Persia agents' routing algorithm. Figure 8 and 9 show the comparison of two algorithms in Random-Small and Kobe maps.

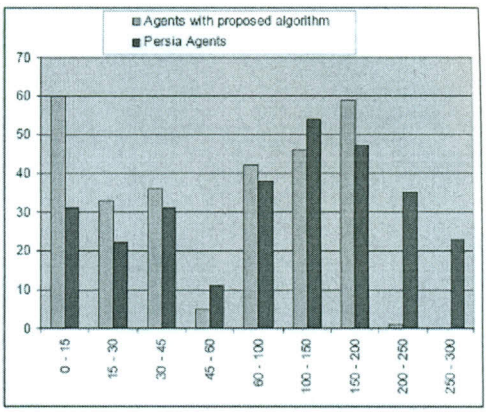

Figure8.The proposed algorithm in comparison with Persia agents' algorithm from exploration view in Kobe map.

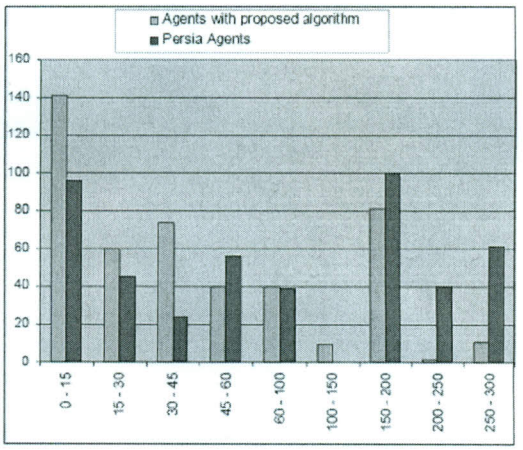

Figure9.The proposed algorithm in comparison with Persia agents' algorithm from exploration view in Random-Small map.

IV. CONCLUSION

In this paper we proposed an exploration algorithm for police agents which we applied the imitation of traffic management solution in human society to the agent society in rescue simulation environment. We did this by using distributed learning automata. Simulations show that proposed algorithm can explore more spaces in the environment at the same time in comparison with other algorithms which use shortest path algorithms. In contrast with other algorithms, the peak of exploration in proposed algorithm is in the beginning of simulation. These make the proposed algorithm be a more suitable algorithm for search and exploration in dynamic and non-deterministic environments.

REFERENCES

1. H. Kitano, S. Tadokoro, et al. RoboCup-Rescue: Search and Rescue in Large-Scale Disasters as a Domain for Autonomous Agents Research, Proc. of IEEE SMC. 1999.
2. M. R. Khojasteh, A. Kazimi and Z. Ghaseminik, "Persia 2006, Towards a Full Learning Automata-Based Cooperative Team", *Team Description Paper*, 2006.

3. 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.
4. M. Asghari, B. Masoumi and M. R. Meybodi, "Improving the Exploration Power of Rescue Agents Using Ant Colony and Learning Automata", *The first Symposium of RoboCup Iiranopen2009*, Iran, 2009.
5. K. S. Narendra and M. A. L. Thathachar, "Learning Automata: An Introduction ", Prentice Hall, Inc., 1989.
6. M. A. L. Thathachar, P. S. Sastry, "Varieties of Learning Automata: An Overview", IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics, Vol. 32, No. 6, pp. 711-722, 2002.
7. H. Beigy and M. R. Meybodi, "A New Distributed Learning Automata for Solving Stochastic Shortest Path Problem", Proceeding of the Sixth International Joint Conference on Information Science, Durham, USA, pp. 339-343, 2002.
8. M. R. Khojasteh and H. Heidari, "Persia 2005 Team Description. Team Description Paper", 2005.
9. A. Kleiner, M. Brenner, T. Brauer, C. Dornhege, M. Gobelbecker, M. Luber, J. Prediger, J. Stuckler,and B. Nebel, "Successful Search and Rescue in Simulated Disaster Areas", Institute for Informatic, University at Freiburg, Germany.
10. T. Morimoto, "How to Develop a RoboCupRescue Agent for RoboCupRescue Simulation System", version 0, 1st edition.