Improving the Exploration Power of Rescue Agents Using Ant Colony and Learning Automata

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

[#1] *Islamic Azad University, Miandoab Branch, Iran,*

[*2] *Islamic Azad University, Qazvin Branch, 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 multiagent system's related ideas and techniques. Hence, the world RoboCup competitions is hold each year and the used ideas and techniques are evaluated in the form of different teams. Almost all of participated teams in the world RoboCup competitions use the shortest path algorithms (like dijkstra algorithm) for path planning. 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. The main purpose of each team is to minimize the humanoid injuries. 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 road and the status of explored areas to the fire brigade and ambulance agents. Hence, in the search phase it is suitable that we have an algorithm with a high exploration power, which is able to search and explore the largest area in the shortest time. We proposed a hybrid algorithm, which uses ant colony and learning automata and gives more exploration power to the agents**.

Keywords— **search and exploration, ant colony optimization, 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. 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. Hence, in the search phase it is suitable that we have an algorithm with a high exploration power, which is able to search and explore the largest area in the shortest time. In this paper, we propose a new hybrid algorithm, which uses ant colonies and learning automata that can be used to more flexibility and exploration power to the agents.

The content of this article is as followed: First, in section 1 the ant colony optimization is introduced briefly. The learning automata is stated in section 2. The section 3 introduces the proposed algorithms. In section 4, we will evaluate the proposed algorithms in the learning and non-learning cases and finally we will compare the proposed algorithms with the base algorithm (as benchmark) in section 5.

II. Ant Colony Optimization

Ant Colony Optimization, the first time proposed by Marko Dorigo [2], is a new class of natural algorithms inspired by the foraging behaviour of natural ant colonies [3]. The first algorithm was aiming to search for an optimal path in a graph; based on the behavior of ants seeking a path between their colony and a source of food. The original idea has since diversified to solve a wider class of Numerical problems, and as a result, several problems have emerged, drawing on various aspects of the behavior of ants. The field of ACO studies artificial Ant Systems for solving discrete optimization problems. For example, in the case of the finding the shortest path as an optimization problem, the problem description represented as a graph. The construction of the shortest path is motivated by how real ants find the shortest path from their nest to a food source and back. By analogy with the pheromone trail real ants use, each graphs' edge, connecting node with node, has a variable associated with it. This variable represents the amount of pheromone on the edge, while moving ants add pheromone to the edges on their way. They move according to a probabilistic decision policy based on the amount of pheromone trail they "smell" on the graph's edges. Positive feedback is thus implemented by reinforcing a trail on those edges. To avoid some premature convergence this pheromone trail evaporates over time and ants' transitions to other nodes in the graph happen stochastic. The edges, which have more amount of pheromone, have high probability to be selected as path by ants.

-Path selection in ACO

At a time instance t, each node in graph has a number of ants that choose the next node to go to with a probability that is a function of the node distance and of the amount of pheromone trail present on the connecting edge. Every edge chosen will be updated with a value depending on the length of the tours. To avoid some premature convergence, this pheromone trail slowly evaporates. For the same reasons state transitions are stochastic. Communication is handled by the update of the pheromone trail, which is locally available for the other ants. An ant will move from node i to node j with probability $p_{i,j}$:

$$p_{i,j} = \frac{[\tau_{i,j}]^{\alpha}.[\eta_{i,j}]^{\beta}}{\sum_{j_k=1}^{n}[\tau_{i,j_k}]^{\alpha}.[\eta_{i,j_k}]^{\beta}} \tag{1}$$

$\eta_{i,j} = 1 / d_{i,j}$ where $d_{i,j}$ is the distance between node i and node j and $\tau_{i,j}$ is the intensity of the trail on edge (i,j). α is a parameter to control the influence of $\tau_{i,j}$ and β is a parameter to control the influence of $\eta_{i,j.}$

-Pheromone update:

After the ants in the system ended their tours, the trail values of every edge (i, j) are updated as follow:

$$\tau_{i,j}(t+1) = \rho\tau_{i,j}(t) + \Delta\tau_{i,j}(t,t+1) \tag{2}$$

where ρ is a user-defined parameter called evaporation coefficient such that $0 \leq \rho \leq 1$ and

$$\Delta\tau_{i,j}(t,t+1) = \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}(t,t+1) \tag{3}$$

$\Delta\tau_{i,j}^{k}(t,t+1)$ is the quantity of length of trail substance laid on edge (i,j) by the kth ant between time t and $t+1$. The m parameter is total number of ants in the system. For more information, the reader may refer to [3].

III. Learning Automata

Learning Automata are adaptive decision-making devices operating on unknown random environments [4]. 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) [5]. 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 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.

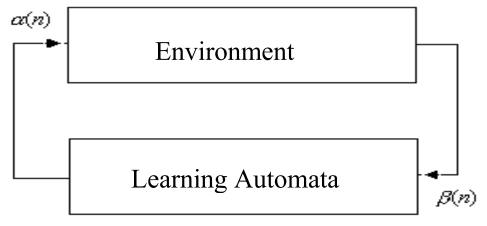

Figure 1. Learning automata - environment

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 $ci = [\beta \mid \alpha_i], i:1..r$. 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:

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)] \tag{4}$$

$$p_j(n+1) = (1-a)p_j(n) \tag{5}$$

-In undesired (failure) case:

$$p_i(n+1) = (1-b)p_i(n) \tag{6}$$

$$p_j(n+1) = (b/r-1) + (1-b)p_j(n) \quad \forall j \quad j \neq i \tag{7}$$

a, 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 [5].

IV. Hybrid search and explore algorithm

The proposed algorithm for search and explore uses ant colony and learning automata in a hybrid case. In order to select a path from different paths, in this algorithm the mechanism of ant colony is used and for pheromone update, the learning automata's reward-penalty method is used. The aim of algorithm is to explore and search in the environment and to avoid of repeated searches. Each edge, which is selected according to ant colony mechanism, gets a penalty according to learning automata rules and the other edges (which have not been selected) get rewards. The probability of visited edges selection in the future is decreased and the non-visited edges will have a more chance to be selected in the future. Therefore, each agent tries to avoid of visiting the edges, which has visited before. So the repelling ant colony mechanism is used which updates the pheromone according to learning automata's reward-penalty rule. The selection of next node (V_{i+1}) for V_i in this algorithm is:

```
Suppose:
$d_{ij}$ = the length of edgei-j (edge between nodes i and j)
$S_{ij}$ = cost of moving on edgei-j (from node i to node j)
1-For each edge Ej in Vertex i
    Set $S_{ij} = 1/d_{ij}$
2-Compute the probability value for all adjacent edges to
    node i based on $S_{ij}$ values.
3-Select one of the adjacent edges to node i randomly.
4-For each edge $E_j$ in Vertex i
  Begin
      If edge $E_j$ is not selected
      Then
        Update pheromone on edge $E_j$ as reward according
        to equation 6
      Else
        Make its pheromone evaporation as a penalty
        according to equation 7
  End
```

V. Evaluation

The graph we used for evaluation is a part of Kobe city graph, which has 293 nodes. The target selection method for agents in all algorithms and in all situations is the same. The recorded data for each algorithm is the result of first 60-cycle simulation. At first a sample of the algorithm performance, which uses the shortest path to reach the target, is proposed. For comparison, table1 and Figure2 is the exploration algorithm performance of Persia team in the first 60 cycle of simulation [6],[7]. Since this algorithm uses the shortest paths to go to the target, only thus nodes are visited which are located in shortest path. After 30 time cycles, the 37% of roads are explored and cleaned. After 60 time cycles, 55% of roads are cleaned too, and 133 roads are which not explored yet.

A. The proposed algorithm in non-learning case

To determine the effect of learning in proposed algorithm, first we evaluated this algorithm in non-learning case. In this case, the next node selection is done on the base of probability, which is constant during simulation. The performance function of this algorithm, in the first 60-time cycle of simulation is in table 2.

According to Table 2 and Figure 3, After a 30 time cycle, 61% of the roads are Explored and cleaned, after a 60 time cycle, 94% of roads were explored and eighty roads were left without exploration. When the edge selection probability is constant during simulation, 3800 selection of 5044 selection is optimal and 1344 selections are non-optimal, then in approximately 74% of cases, the edges are selected with low costs (selected edges belong to the shortest path between source and target).

TABLE 1. NUMBER OF NODES VISITED BY PERSIA AGENTS IN FIRST 60 SECOND OF SIMULATION

Time	Number of visited nodes	Time	Number of visited nodes	Time	Number of visited nodes	Time	Number of visited nodes
1	0	16	72	31	112	46	129
2	27	17	79	32	112	47	129
3	27	18	79	33	112	48	130
4	27	19	95	34	112	49	130
5	27	20	95	35	116	50	147
6	29	21	95	36	125	51	147
7	29	22	96	37	125	52	147
8	29	23	96	38	125	53	150
9	29	24	101	39	125	54	150
10	29	25	101	40	125	55	150
11	29	26	106	41	125	56	155
12	29	27	106	42	125	57	155
13	29	28	106	43	125	58	156
14	55	29	106	44	125	59	156
15	72	30	**106**	45	129	60	**159**

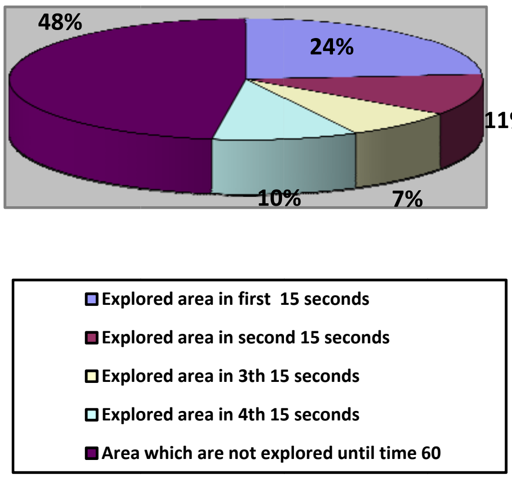

Figure 2. Exploration rate of algorithm, which were used in Persia team in first 60 second of simulation.

B. The proposed algorithm with learning ability

In the proposed algorithm, selection of next edge is done on the base of probability. Algorithm uses reward-penalty method, so that each selected edge takes penalty and other edges, which are not selected, take rewards, so these edges (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 learning scheme we used is reward-penalty, which uses the same value for reward, and penalty. The value for parameter *b* is *0.1.* Performance function of this algorithm is as table 3.

According to the figure 4, this algorithm acts in an exploding manner at the beginning because there is no absolute selection of an edge in a path. After 30 seconds, 85% of the roads and after 60 seconds 98% of the roads were explored and cleaned and after 63 seconds, only three roads were left without exploration.

TABLE 2. NUMBER OF NODES VISITED BY PROPOSED ALGORITHM IN NON-LEARNING CASE IN FIRST 60 SECOND OF SIMULATION

Time	Number of visited nodes		Time	Number of visited nodes		Time	Number of visited nodes		Time	Number of visited nodes
1	0		16	144		31	187		46	241
2	22		17	144		32	187		47	241
3	41		18	144		33	195		48	245
4	58		19	144		34	213		49	245
5	73		20	144		35	228		50	247
6	106		21	144		36	229		51	255
7	106		22	155		37	233		52	264
8	106		23	155		38	235		53	271
9	116		24	158		39	236		54	271
10	116		25	163		40	239		55	271
11	117		26	173		41	239		56	274
12	117		27	173		42	239		57	275
13	137		28	173		43	240		58	275
14	142		29	180		44	240		59	275
15	142		30	**180**		45	240		60	**274**
Total number of nodes = 293 number of nodes which were left without visiting =19										

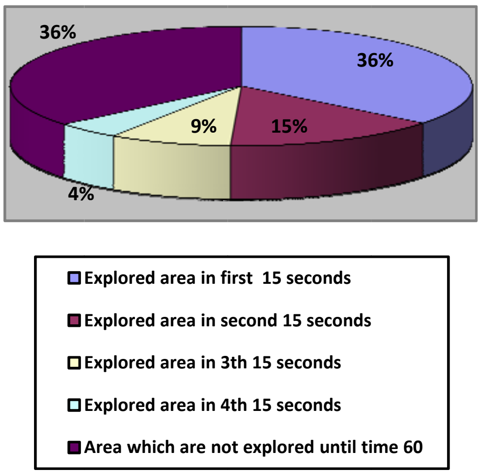

Figure 3. Exploration Rate for proposed algorithm in non-learning case

TABLE 3. NUMBER OF NODES, WHICH PROPOSED ALGORITHM COULD EXPLORE IN FIRST 60 SECOND OF SIMULATION WHEN LEARNING ALGORITHM IS USED.

Time	Number of visited nodes		Time	Number of visited nodes		Time	Number of visited nodes		Time	Number of visited nodes
1	0		16	185		31	255		46	281
2	18		17	185		32	255		47	281
3	38		18	185		33	255		48	281
4	64		19	195		34	257		49	281
5	81		20	195		35	257		50	281
6	88		21	209		36	259		51	281
7	96		22	229		37	259		52	281
8	119		23	238		38	271		53	281
9	129		24	246		39	271		54	281
10	141		25	246		40	271		55	281
11	160		26	246		41	271		56	281
12	175		27	246		42	276		57	284
13	177		28	246		43	276		58	284
14	185		29	247		44	280		59	284
15	185		30	**247**		45	281		60	**285**
Total number of nodes = 293 number of nodes which were left without visiting =8										

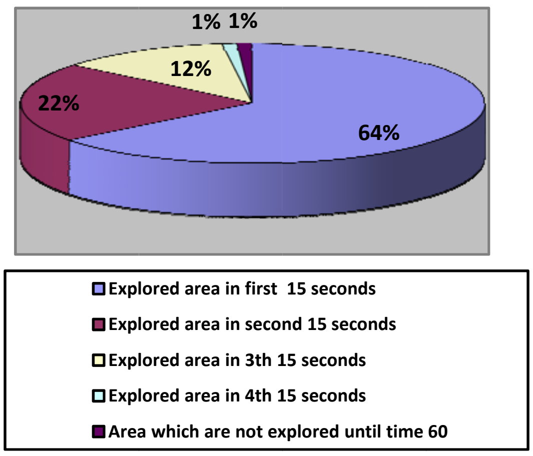

Figure 4. Exploration rate of proposed algorithm with learning ability.

VI. CONCLUSION:

In this paper we proposed two algorithms for search and exploration in Rescue simulation environment: an algorithm without learning which uses constant probability for edge selection and other with learning, which uses ant colony and learning automata in a hybrid case. The algorithm, which we used as benchmark, explored the target area in approximately 100 seconds while the proposed algorithm in non-learning case explored the same area in 60 seconds. The hybrid algorithm explored the same area in 38 seconds. In contrast to other similar algorithms, peak of exploration in the proposed algorithms is in the beginning of simulation. In other word, the proposed hybrid algorithm acts in an exploding manner at the beginning of simulation. These make the proposed algorithm be a more suitable algorithm for search and exploration in dynamic and non-deterministic environments.

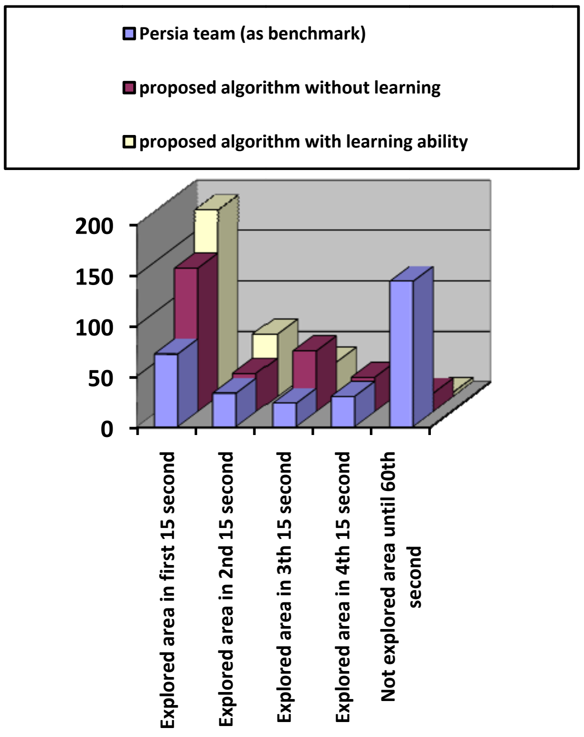

Figure 5. Comparison of proposed algorithm with a benchmark algorithm.

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. Dorigo, G. DiCaro and M. L. Gambardella, “Ant Algorithms for Discrete Optimization”, Artificial. Life, vol. 5, no. 2, pp. 137–172, 1999.

[3] M. Dorigo and T. Stützle, “Ant Colony Optimization”, Cambridge, MA: The MIT Press.2004.

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

[5] 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.

[6] M. R. Khojasteh and H. Heidari, “Persia 2005 Team Description. Team Description Paper”, 2005.

[7] M. R. Khojasteh, A. Kazimi and Z. Ghaseminik, “Persia 2006, Towards a Full Learning Automata-Based Cooperative Team. Team Description Paper”, 2006.

[8] 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 Informatics, University at Freiburg, Germany, 2005.

[9] T. Morimoto, “How to Develop a RoboCup Rescue Agent for RoboCup Rescue Simulation System”, version 0, 1st edition, 2002.

[10] 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.