Utilizing Learning automata and Entropy to Improve the Exploration Power of Rescue Agents

Behrooz Masoumi
Department of Computer Engineering
Islamic Azad University, Qazvin Branch
Qazvin, Iran.
Masoumi@Qiau.ac.i

Mostafa Asghari
Department of Computer Engineering
Islamic Azad University,
Mianoab Branch,
m_asghary86@yahoo.com

Mohammad Reza Meybodi
Department of Computer Engineering
Amirkabir University of Technology,
Tehran, Iran
mmeybodi@aut.ac.ir

***Abstract*—Rescue Simulation System is an example of multi-agent systems in which we encounter many challenges. One of these challenges is to having Tradeoff between exploration and exploitation in path planning phase. In this paper we present an exploration method based on variable structure *S* model learning automaton which uses the entropy of action's probability vector as a criteria to give reward or to penalize its selected action. This method can leads agents to establish a logical balance between exploration and exploitation too. The results show that the proposed method has good performance from both exploration and acquired final score point of view in rescue simulation system.**

Keywords- Search and Exploration; Learning Automata; Entropy; Rescue Simulation.

I. Introduction

RoboCup Rescue Simulation System is a large-scale, real-time and multi-agent simulation of urban disaster [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.

The simulated disaster environment consists of a map of a city (or of a virtual or imagined city). Basic unit of a map is a node. Nodes are connected by nodes, and buildings can be found opening off nodes. The scenario begins by assuming that there has been an earthquake in the city. At the beginning of a simulation, a number of buildings may have collapsed, possibly with human buried inside. Building collapse may cause road blockage and some buildings may have ignited. In order to save lives and minimize damage in simulated disaster environment, rescue agents are developed and programmed to mitigate disaster after-effects [2].

The simulated environment is highly dynamic and only partially observable by a single agent. Agents have to plan and decide their actions asynchronously in real-time. There are tree types of rescue agents with specific capabilities in rescue simulation system: ambulance agents are able to recover buried civilians, and transfer them to refuges where they can be tended to; fire fighter agents are able to extinguish fires, and police agents are able to clear blocked roads. An example of such simulated environment is illustrated in Figure 1. In a partially observable environment like Rescue simulation system, exploration is the key means for agents to enlarge their knowledge. It is especially important to find injured civilians as quickly as possible without losing time by redundant and repeated exploration of the same area by the agents.

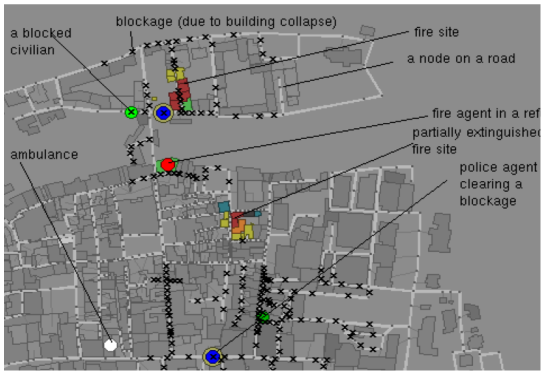

Figure 1. A Simulated disaster environment in Rescue Simulation system

The main goal of the agents is to rescue more civilians [1]. To aim this, and consequently to achieve high score, it is necessary that police agents can search the environment fast as possible as, and open the blocked roads so that the ambulances and fire brigades be able to rescue the injured civilians and extinguish the burning buildings. This is the duty of police agents. Most of the participated teams in the world Robocup competitions use the shortest path algorithms (like DIJKSTRA algorithm) to move from target to their destination [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 hybrid algorithm, which uses ant colonies and learning automata that can be used to give more flexibility and exploration power to the agents.

In this paper we propose a new method based on variable structure, *S-model* learning automaton and the entropy concept in order to enhance the efficiency of police agent in search and exploration of the environment. Entropy is a significant concept in the thermodynamics, representing the degree of disorder in a thermodynamic system [5], [6]. In this paper the concept of entropy has been used to improve the learning process. Furthermore, we have placed a variable structure, *S-model* learning automaton in each node of simulated disaster environment. When an agent wants to move from one node to another node, at each node, the corresponding learning automaton is activated. If the set of selected actions of learning automata in node *i* leads to the next node *j* then the selected action of the learning automaton for node *i* is updated based on the entropy of the probability vector of the learning automaton in the node *j*. Reminder of this paper is organized as follow: first in section II we have introduced the learning automata briefly. The proposed method is presented 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 [7]. 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 2 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) [8], [9]. In the following, the variable structure learning automata is described.

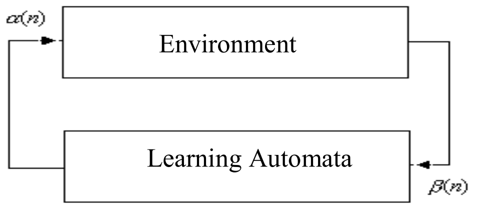

Figure 2. Learning automata - environment

Variable structure learning automata can be shown by a quadruple *{ α , β, p, T}* where *α={α₁, α₂, ..., α_r}* which is the set of actions of the automaton, $\beta=\{\beta_1, \beta_2, \ldots, \beta_m\}$ is its set of inputs, $p=\{p_1, \ldots, p_r\}$ is probability vector for selection of each action, and $p(n+1) = T[\ (n),\ \ (n), p(n)]$ is the learning algorithm. If $\beta=\{0,1\}$, then the environment is called *P-Model*. If β belongs to a finite set with more than two values, between 0 and 1, the environment is called *Q-Model* and if β is a continuous random variable in the range [0, 1] the environment is called *S-Model*. Let a *VSLA* operate in an *S-Model* environment. A general linear schema for updating action probabilities when action *i* is performed is given by:

$$\begin{aligned} p_i(n+1) &= p_i(n) + a(1-\beta_i(n))(1-p_i(n)) - b\beta_i(n)p_i(n) \\ p_j(n+1) &= p_j(n) - a(1-\beta_i(n))p_j(n) + \\ & b\beta_i(n)\left[\frac{1}{r-1} - p_j(n)\right] \forall j \quad j \neq i \end{aligned} \tag{1}$$

where *a* and *b* are reward and penalty parameters. When *a=b*, the automaton is called $S\text{-}L_{R\text{-}P}$. If *b=0* and *0<b<<a<1*, the automaton is called $S\text{-}L_{R\text{-}I}$ and $S\text{-}L_{R\text{-}\varepsilon P}$, respectively. The overall operation of learning automaton is summarized in Figure 3.

```
VSLA Algorithm
Initialize p to [1/r, 1/r,…,1/r] where r is the number of actions
    while not done
        Select an action i based on the probability vector p
        Evaluate the action and return a reinforcement signal β
        Update the probability vector using the learning algorithm
    end while
```

Figure 3. Pseudo code of variable-structure learning automaton

III. The entropy Concept

Entropy is a significant concept in the thermodynamics, representing the degree of disorder in a thermodynamic system that is played an important role in various fields of computer science, such as coding theory, learning, compression, and others [5], [6]. Shannon has introduced this concept into the information theory, by the name of "information entropy". Entropy, in its basic, indicates a measure of uncertainty rather than a measure of information. More specifically, the information entropy is a case of the entropy of random variables defined as follows [6]:

$$H(X) = -\sum_{X\in\chi} P(X)\log(P(X)) \tag{2}$$

where *X* represents a random variable with set of values χ and probability mass function *P(X)*. Entropy is always a positive value and can change bases freely as $H_b(X) = \log_b(a).H_a(X)$. For the random variable *X* and with *Xlog(X)* tending to zero as *X* tends to zero. Entropy measures the uncertainty inherent in the distribution of a random variable.

IV. The Proposed Algorithm

In the proposed algorithm, there is a variable structure *S-model* learning automaton on each node of environment 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. At the initialization phase, all of the automaton's actions are given the same probability values. When an agent wants to pass a node, the corresponding learning automaton is activated and proposes one of the outgoing edges from current node to the agent as a part of its path. The selection phase for each automaton is done based on probability rule. If the selected action of automaton leads the agent to reach to the target node, automatons' selected action get rewards, otherwise the active learning automaton uses the entropy of the action's probability at the next node in the path as a criteria for reward or penalty. Entropy value for action's probability in a particular node shows that how much the information about target node is uncertain. High values for entropy means high uncertainty in information about target. This means that whenever an automaton has high entropy value for its action's probability, it doesn't have useful information about target and selects its actions most randomly and having low entropy value in a particular node means that corresponding learning automaton has useful information about target and selects its action with high probability.

Assume that $P(n)=\{p_1, p_2,...,p_r\}$ be the probability values for a learning automaton with r action in node n, the entropy value for that automaton's action's probability is calculated as follow:

$$H(n)=\sum_{i=1}^{r} p_i^n \log\left(p_i^n\right) \tag{3}$$

Entropy has its maximum value when all the actions have equal probabilities of selection and has value zero (its minimum) when the action probability vector is a unit vector. In order to be able to use entropy as a reinforcement signal for *S-Model* variable structure learning automata, the entropy needs to be rescaled in the range of [0,1]. Suppose that agent is in node n and its learning automaton that is LA (n*)*, leads the agent to node *n'*. In this case, reinforcement signal, $\beta(n)$, as given in using the following formula:

$$\beta(n)=H(n')/(MaxH(n'))^k \tag{4}$$

where *MaxH(n')* is maximum entropy in node *n* for the agent defined as:

$$Max(H(n'))=\sum_{j=1}^{r(k)}\frac{1}{r(k)}\log(\frac{1}{r(k)})=\log_2^{r(k)} \tag{5}$$

In (4) the k parameter is used to make a balance between exploration and exploitation. Having high values for k leads the algorithm towards exploration and low values for k causes the automatons penalize their actions and leads the algorithm towards exploitation. This method acts in a manner so that at first agents are intended to have more exploration in environment. As the time pass and the parameter changes, agents want to use their learned information and this leads them towards exploitation. Suppose that agent k be in node n, and learning automata which is corresponded to n, *LA(n)*, takes the agent into node *n'*, in this case reinforcement signal is determined as follow:

$$\beta(n)=\begin{cases} 0 & \text{if } n' \text{ is Goal} \\ H(n')/Max(H(n'))^k & \text{otherwise} \end{cases} \tag{6}$$

Since the input for *S-model* learning automaton must be in [0, 1], using entropy as input for learning automatons gives us opportunity to use learning automaton in systems like rescue simulation system. On the other hand using entropy values as criteria for reward or penalty in learning automatons enables having a logical balance between exploration and exploitation.

Another important point is that, when a learning automaton is activated, it gives reward or penalty to its selected action, according to (6). Consequently, depending on β value (entropy value) some edges will have lower chance to be selected in the future and some will have more chance to be selected. Giving reward or penalty to edges, will cause all of the outgoing edges have chance to be selected by the agents as part of their path. As time pass, Giving reward or penalty to the edges, leads the algorithm from exploration toward exploitation. Selection of next node (V_{i+1}) for each V_i is as figure 4.

```
The proposed Algorithm
Initialize the parameters i.e. (a=0.05, k =1).
2. Initialize the probability vector for all
   automatons actions(at first all actions have the
   same probability value).
3. Select one of the outgoing edges from Vi randomly.
4. Compute $\beta_i$ for Vi+1 and apply to the Eq. 1.
5. for all adjacent Edges (Ej) to Vi do
     If Ej has been selected as path
       Reward or penalize learning automata in Ej
        according to Eq.1
     end for
```

Figure 4. Pseudo code for proposed algorithm

As mentioned in the previous sections, the learning scheme which is used in this algorithm is variable structure, *S-model* with linear reward-penalty (*S-L*$_{RP}$). Value for Parameter *a* is computed on the base of try and error method. We have used the entropy value as value for parameter β.

V. Evaluation

To evaluate the proposed algorithm, Kobe map with 292 nodes and 381 edges (default map for rescue simulation system) is used. The Persia agents, which use shortest path algorithm (DIJKSTRA algorithm) to reach their target, are used as benchmark to evaluate our proposed algorithm [3]. It is assumed that the target selection method for agents in our algorithm and benchmark algorithm (Persia) in all situations is the same. For comparison, Table 2 indicates the exploration performance of proposed algorithm and Persia agents' algorithm during the simulation with used maps. Exploration rate for proposed algorithm and Persia agents during the simulation are as Figure 4 and Figure5.

TABLE I. THE NUMBER OF NODES VISITED BY AGENT WHICH USES OUR ALGORITHM IN COMPARISON WITH PERSIA AGENT THE DURING SIMULATION WITH A MAP WITH 293 NODES.

Time / Algorithm	0 – 15	15 – 30	30 - 45	45 - 60	60 - 100	100 -150	150 -200	200 - 250	250 - 300
Persia Agent	*31*	*22*	*31*	*11*	*38*	*54*	*47*	*35*	*23*
Proposed Method	*60*	*42*	*35*	*9*	*57*	*40*	*11*	*20*	*14*

The exploration rate for Persia algorithm and proposed algorithm are shown respectively in Figure 5 and 6.

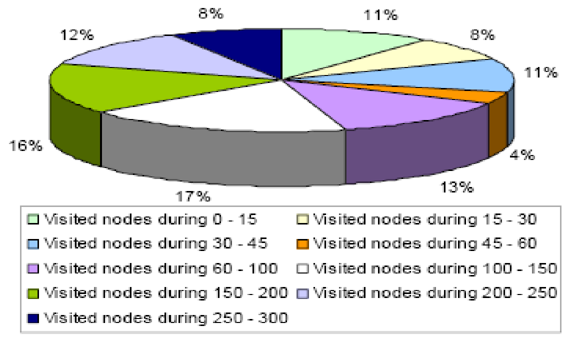

Figure 5. Exploration rate for Persia police agents in used map

In the proposed method, the next node selection is done on the basis of probability which varies during simulation. The performance function of this algorithm during the simulation is as follows:

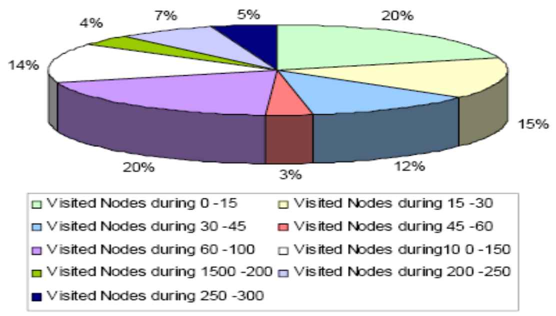

Figure 6. Figure5. Exploration rate for our agents in used map

In the previous section, Persia teams' routing algorithm and our proposed algorithm are simulated and evaluated separately. To show the efficiency of the proposed algorithm from both exploration view and final score, we compared it with Persia agents' routing algorithm in Figure 7 and 8.

I. CONCLUSION

In this paper we proposed an exploration algorithm for police agents which use variable structure *S-model* learning automata. Using actions' entropy values as learning automatons' input enable agents to balance exploration and exploitation. Simulations show that proposed algorithm can improve the final score and also can explore more spaces in the environment at the same time in comparison with other algorithms which use shortest path algorithms. These make the proposed algorithm be a more suitable and flexible algorithm for search and exploration in dynamic and non-deterministic environments.

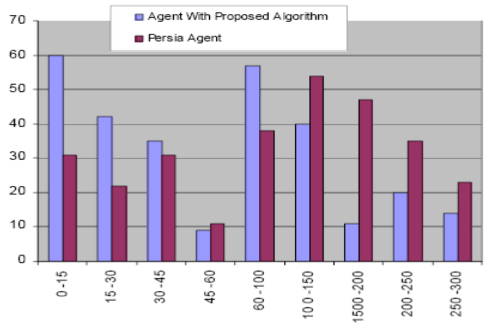

Figure 7. The proposed algorithm in comparison with Persia agents' algorithm from exploration view in used map.

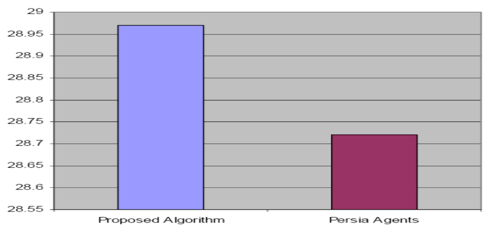

Figure 8. The proposed algorithm in comparison with Persia agents' algorithm from final score view in used map.

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] 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*.

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

[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] E. H. Lieb and J. Yngvason, "The Physics and Mathematics of the Second Law of Thermodynamics," *Physics Report,* vol. 310, pp. 1-96, 1999.

[6] Z. Dianhu, F. Shaohui and D. Xiaojun, "Entropy – A Measure of Uncertainty of Random Variable," Systems Engineering and Electronics, no. 11, pp. 1-3, 1997.

[7] X. Zhuang, "The Strategy Entropy of Reinforcement Learning for Mobile Robot Navigation in Complex Environments," in *the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, 2005, pp. 1742-1747.

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

[9] 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.