RoboCupRescue 2011 – Rescue Simulation League Team Description <BonabRescue (Iran)>

Farshid Faraji, Mohamad Reza Khojasteh, Hekmat Hajizadeh, Reza Moheb Alizadeh, Naser Irani

AI & Robotics Laboratory, Computer Group, Islamic Azad University of Bonab Iran
{faraji, hajizadeh, moheb, irani}@bonabiau.ac.ir
http://www.Bonabiau.ac.ir/BonabRescue/

Abstract. Rescue Simulation System is a multi-agent system in which we encounter many challenges like Tradeoff between exploration and exploitation in path planning phase and finding shortest and optimal path between two nodes. We have used 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. And also we have used Distributed Learning Automata (DLA) to find a policy that determines a path from source node to a destination node with minimal expected cost.

1. Introduction

Some of the main ideas behind our team were derived from an MS thesis on using and investigating Learning Automata [3] in cooperation among agents in a team which could act in a complex multi-agent domain [2][3][4][5]. Some of these ideas were first built in RoboCup Soccer2D Simulation domain at that time [3]. By now we used similar ideas in RoboCup Rescue Simulation domain each year. This year, we have extended our use of Learning Automata (and one of its derivations: Distributed Learning Automata) in much more parts of our simulated team in the domain of RoboCup Rescue Simulation.

In this paper, we have used Learning Automata and Distributed Learning Automata [1][2][3][4][5] as our machine learning methods in exploration and path planning phase of our agents' development.

The main goal of the agents is to rescue more civilians [6]. To aim this, and consequently to minimize damage, it is necessary that agents can search the environment fast as possible as. 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 [7]. 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 BonabRescue 2008 [8] we proposed an algorithm, which uses ant colonies and learning automata that can be used to give more flexibility and exploration power to the agents.

This year we have used a new method based on variable structure, S-model learning automaton and the entropy concept in order to enhance the efficiency of agents in search and exploration of the environment. Entropy is a significant concept in the thermodynamics, representing the degree of disorder in a thermodynamic system [9][10]. 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*.

The next algorithm that we have used in our team is based on a Distributed Learning Automata (DLA) to find a policy that determines a path from source node to a destination node with minimal expected cost (length). Here DLA is applied to the shortest path problem. In order to compute the probability that a path being the shortest, a distributed learning automaton is constructed from the given input graph and each learning automaton in the DLA updates its action probability vector using L_{R-I} reinforcement scheme until the shortest path is found.

2. LEARNING AUTOMATA

Learning Automata are adaptive decision-making devices operating on unknown random environments [11]. 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

Learning Automata can be classified into two main families: fixed structure learning automata and variable structure learning automata (VSLA) [1].

Variable structure learning automata can be shown by a quadruple { α , β, p, T } where α={α1, α2, ..., αr} which is the set of actions of the automaton, β={β1, β2,…, βm} is its set of inputs, p={p1, ..., pr} is probability vector for selection of each action, and p(n +1) = T[(n), (n), p(n)] is the learning algorithm. If β={0,1}, then the environment is called PModel. 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 a SModel environment. A general linear schema for updating action probabilities when action i is performed is given by:

$$p_i(n+1) = p_i(n) + a\big(1-\beta_i(n)\big)\big(1-p_i(n)\big) - b\beta_i(n)p_i(n)$$
$$p_j(n+1) = p_j(n) - a\big(1-\beta_i(n)\big)p_j(n) + b\beta_i(n)\left[\frac{1}{r-1} - p_j(n)\right] \quad \forall j \; j \neq i \qquad (1)$$

where a and b are reward and penalty parameters. When a=b, the automaton is called S-LR-P. If b=0 and 0<b<<a<1, the automaton is called S-LR-I and S-LR-εP, respectively.

3. Distributed learning automata

Distributed learning automata is a network of automata which collectively cooperate to solve a particular problem. A DLA can be modeled by a directed graph in which the set of nodes of graph constitute the set of automata and the set of outgoing edges for each node constitutes the set of actions for corresponding 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. An example of DLA is given in figure 1, in which every automaton has two actions. If automaton A_1 selects action as, then automaton A_3 will be activated. Activated automaton A_3 chooses one of its actions which in turn activates one of the automata connected to A_3. At any time only one automaton in the network will be activated. Formally, a DLA with n learning automata can be defined by a graph (A, E), where A = {A1, Az, ... ,A,) 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 α_j of automaton A_i. Let action probability vector for learning automaton A_j is represented by p^j where a component of P^j_m denotes the probability of choosing action α_m, that is, the probability of choosing edge (j, m).

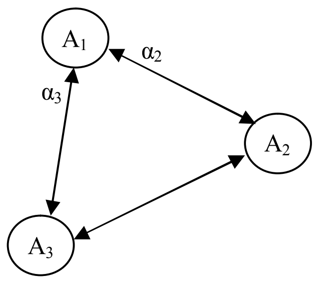

Fig 1. Distributed Learning Automata

4. 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 [9], [10]. 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 [10]:

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

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 Hb(X) = logb (a).Ha(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.

5. Exploration based on learning automaton and the entropy

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)={p1, p2,...,pr} 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, β(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:

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

In formula (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:

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

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. 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.

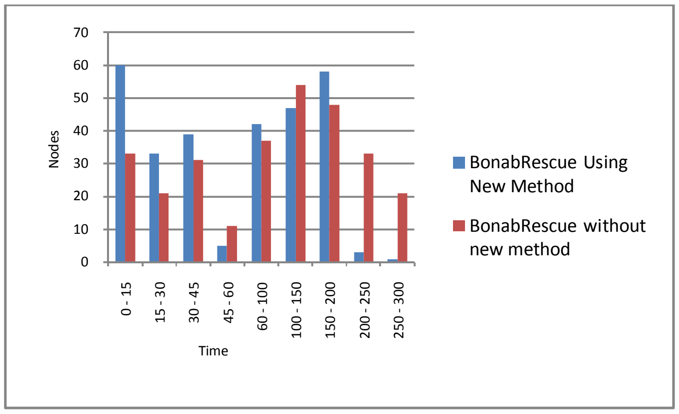

Fig 2. The new algorithm in comparison with BonabRescue without using new algorithm from exploration view in Kobe map.

Simulations show that new 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 new algorithm is in the beginning of simulation. These make the new algorithm be a more suitable algorithm for search and exploration in dynamic and non-deterministic environments.

6. Path planning using Distributed learning automata

At first a network of learning automata which is isomorphic to the input graph is created. In this network each node is a learning automaton and each outgoing edge of this node is one of the actions of this learning automaton. The output of DLA is a sequence of actions that represents a particular path in the graph. At the first step, source automaton (corresponding to the source node in the graph) A_s chooses one of its actions (as a sample realization of its action probability vector), say action α_m. This action activates automaton A_m on the other end of *edge(s , m)*. The process of choosing an action and activating an automaton is repeated until destination automaton A_d is reached or for some reason moving along the edges of the graph is not possible or the number of visited nodes exceeds the number of nodes in the graph. After A_d is reached, length of the traversed path, (L_{π_i}),is computed and then compared with a quantity called dynamic threshold, T_k. Depending on the result of the comparison all the learning automata (except the destination learning automaton) along the traversed path update their action probabilities. Updating is done in direction from source to

destination or vice versa. If length of the traversed path is less than or equal to the dynamic threshold then all learning automata along that path receive reward and if length of the traversed path is greater than the dynamic threshold or the destination node is not reached, then activated automata receive penalty.
The process of traveling from the source learning automaton to the destination learning automaton is repeated until the stopping condition is reached which at this point the last traversed path is the path which has the minimum expected length among all paths from the source to the destination. The dynamic threshold at time K > 1 is defined as:

$$\Delta T = \frac{1}{k-1}\sum_{i=1}^{k-1} l_i \tag{7}$$

Where l_i is the length of traversed path at iteration *i*. The algorithm stops if the product of the probability of choosing the edge of the traversed path, called path probability, is greater than a certain threshold.

7. Conclusion and Future work

In this article we focused on path planning and environment exploration based on learning automata. Using actions' entropy values as learning automatons' input enable agents to balance exploration and exploitation. Also implementing Distributed Learning Automata in path planning phase enable agents to find shortest path with a probability as close as to unity by proper choice of the parameters.

8. References

1. Narendra K.S. and Thathachar M.A.L., Learning Automata: An Introduction, Prentice Hall, Inc., 1989.
2. Khojasteh M. R. and Meybodi M. R., Using Learning Automata in Cooperation among Agents in a Team, 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, pages 306-312, Covilhã, Portugal, December 5th-8th, 2005.
3. Khojasteh M. R., Cooperation in multi-agent systems using Learning Automata, M.Sc. thesis, Computer Engineering Faculty, Amirkabir University of Technology, May 2002.
4. Khojasteh M. R. and Meybodi M. R., Evaluating Learning Automata as a Model for Cooperation in Complex Multi Agent Domains, Accepted in 10th International Symposium of RoboCup, to be held in June 2006 in Bremen, Germany.
5. Khojasteh M. R. and Meybodi M. R., Learning automata as a model for cooperation in a team of agents, Proceedings of the 8th annual CSI computer conference (CSICC' 2003), pages 116-125, Mashhad, Iran, Feb. 25-27, 2003.

6. 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.
7. 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.
8. F.Faraji, M. R. Khojasteh, M. Asghari and M. Kafshnochi. The Bonab Robotics 2008 RoboCup Rescue Simulation Team Description, Julay2008, Suzhou, China.
9. E. H. Lieb and J. Yngvason, "The Physics and Mathematics of the Second Law of Thermodynamics," Physics Report, vol. 310, pp. 1- 96, 1999.
10. 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.
11. 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.