Utilizing Generalized Learning Automata for Finding Optimal Policies in MMDPs

Samaneh Assar [*], Behrooz Masoumi

Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran

Abstract

Multi agent Markov decision processes (MMDPs), as the generalization of Markov decision processes to the multi agent case, have long been used for modeling multi agent system and are used as a suitable framework for Multi agent Reinforcement Learning. In this paper, a generalized learning automata based algorithm for finding optimal policies in MMDP is proposed. In the proposed algorithm, MMDP problem is described as a directed graph in which the nodes are the states of the problem, and the directed edges represent the actions that result in transition from one state to another. Each state of the environment is equipped with a generalized learning automaton whose actions are moving to different adjacent states of that state. Each agent moves from one state to another and tries to reach the goal state. In each state, the agent chooses its next transition with help of the generalized learning automaton in that state. The experimental results have shown that the proposed algorithm have better learning performance in terms of the speed of reaching the optimal policy as compared to existing learning algorithms.

Keywords: Generalized Learning Automata, Multi agent systems, Markov Games.

1. Introduction

Learning Automata (LA) are adaptive decision making devices suited for operation in unknown environments [1]. One of the principal contributions of LA is that a set of decentralized LA is able to control a finite Markov Chain with transition probabilities and rewards are unknown [2]. Recently this result has been extended to the framework of Markov Games, which is a straightforward extension of single-agent Markov Decision Problems (MDPs) to distributed multi agent decision problems [2], [3].In a Markov Game, actions are the result of the joint action selection of all agents while rewards and state transitions depend on these joint actions obeying the Markov property. In multi agent system research, two main perspectives are found in the literature; the cooperative and non-cooperative perspective [4], [5]. In cooperative multi agent systems, the agents pursue a common goal and the agents can be built expect benevolent intentions from other agents. In contrast, a non- cooperative multi agent systems setting has non-aligned goals, and individual agents try to obtain only to maximize their own profits. In multi agent caused

* Corresponding author. Email: Samaneh.Assar@qiau.ac.ir

by the fact that the environment of agent is dynamic and just empirically observable while the environment (there ward Functions and the transition states) is unknown. Hence, the reinforcement learning methods may be applied in MAS to find an optimal policy in MGs In a purely cooperative MG, which is called a Multi agent MDP (MMDP); all agents share the same reward function. In MMDPs, the agents should learn to find and agree on the same optimal policy. In general MGs, an equilibrium point is sought; i.e. a situation in which no agent alone can change its policy to improve its reward when all other agents keep their policy fixed. In [6] will first demonstrate how GLA can help take the correct actions in large unknown multi agent environments. in [7] propose to use Generalized Learning Automata (GLA), which are capable of identifying regions within the state space with the same optimal action, and as such aggregating states. In [8] proposed several learning automata based multi agent system algorithms for finding optimal policies in fully cooperative Markov Games. In the proposed algorithms, Markov problem is described as a directed graph in which the nodes are the states of the problem, and the directed edges represent the actions that result in transition from one state to another. In [3] a learning automata based multi agent systems in which the concepts of stigmergy and entropy are used to enhance the performance of system and used to find optimal policies in Markov Games was proposed. In this paper, a generalized learning automata based algorithm for finding optimal policies in Markov Games is proposed. This algorithm consists of multiple agents that use generalized learning automata in order to optimize their own behavior that can be effectively used to find the optimal policy in Markov games. In the proposed algorithm, the environment of Markov problem is modeled as a directed graph. The nodes of this graph represent the states and directed edges between nodes represent the actions that result in transition from one state to another. Each node of the graph is equipped with a generalized learning automaton.

The actions of each learning automaton are the outgoing edges of corresponding node. The agents move on this graph and in each state, they get help from corresponding generalized learning automaton to choose a desirable action and move to the next state. Based on the path taken by agents and its goodness in terms of speed and cost, the probability vectors of generalized learning automata will be updated. This process is performed in parallel by all agents, and it iterates several times until the path taken by each agent converges to the optimal path. The rest of this paper is organized as follows: Section 2 briefly presents the concept of Generalized Learning Automata. Definitions for Markov Decision Process and Markov Games as well as the concept of solution in them are discussed in section 3. The proposed algorithm and its variations are given in section 4. Section 5 presents the experimental results. Section 6 concludes the paper.

2. Generalized Learning Automata (GLA)

The generalized learning automata (GLA) is an extended learning automaton where there is provision for an extra input. The GLA model, where the automaton can take an additional input, is also useful in many other situations [9]. Based on this input and its own internal state the unit then selects an action. This action serves as input to the environment, which in turn produces a response for the GLA. Based on this response the GLA then updates its internal state [10].Figure 1 shows the general agent learning setup. Each time step a vector x(n) giving a factored representation of the current system state is generated. This vector is given to each individual agent as input.

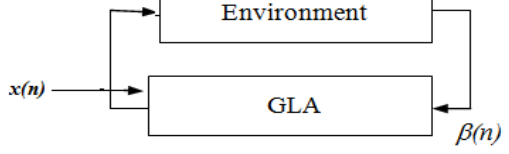

Fig. 1. Learning setup. Each agent receives factored state representation as input. GLA decide action to be performed.

Formally, a GLA is described by the tuple *⟨X, A, β, u, g, U⟩*, where X is the set of inputs to the GLA and A = $\{a_1,..,a_r\}$is the set of outputs or actions the GLA can produce. $B\in[0, 1]$ again denotes the feedback the automaton receives for an action. The vector u is used in conjunction with the probability g to determine the action probabilities, given an input x ∈ X:

$$prob = [a(k) = a_i[u,x] = g(x,a_i,u)] \qquad (1)$$

Where g has to satisfy following conditions:

$$g(x,a_i,u) \geq 0, \forall a_i,u,x$$
$$\sum_{i=1}^{r} g(x,a_i,u) \equiv 1, \forall a_i,x \qquad (2)$$

U is a learning algorithm which updates u, based on the current value of u, the given input, the selected action and response. The earliest algorithm suggested is the REINFORCE algorithm [11] which has the following form:

$$U(t+1) = u(t) + \lambda\beta(t)\frac{\partial \ln g}{\partial u}(x(t),a(t),h(t)) + \lambda k(h(u(t)-u(t)) \qquad (3)$$

Where $h(u)=\lfloor h_1(u_1),h_2(u_2),..h_r(u_r)\rfloor$ with each h_i defined as:

$$h_i(\eta) = \begin{cases} L_i & \eta \geq L_i \\ 0 & |\eta| \leq L_i \\ -L_i & \eta \leq -L_i \end{cases} \qquad (4)$$

In this update scheme, λ is the learning rate and L_i, K_i >0 are constants. The update scheme can be explained as follows. The first term allows the system to locally optimize the action probabilities. The next term uses the h_i(u) functions to keep parameters u_i bounded within predetermined boundaries $[-L_i, L_i]$. In [9] it is shown, that the adapted algorithm described above, converges to local maxima of *f(u) = E[β|u]*, showing that the learning automata find a local maximum over the mappings that can be represented by their internal state in combination with the function g.

The context vectors represent features of objects to be classified and the GLA output represents class labels. We use a factored representation of the state space as the input for the GLA, whereas the outputs are the actions the agent should take. In such a representation the state information is represented as a set of random variables $X=\{X_1, \ldots, X_n\}$, where every state variable X_i can take values in a finite domain *Dom(*X_i*)* and every possible state in the system corresponds to a value assignment $X_i \in Dom(X_i)$ for every state variable, X_i. With T a parameter that represents the temperature to control the exploration. With every action $a_i \in A$ the automaton can perform, it associates a vector u_i. We use the Boltzmann distribution as probability generating function which is written as:

$$g(x,a_i,u) = \frac{e^{\frac{x^T u_i(a_i)}{T}}}{\sum_j e^{\frac{x^T u_{i(a_i)}}{T}}} \qquad (5)$$

3. Control of Markov Game Using Learning Automata

The problem of controlling Single Agent MDPs can be modeled as a network of interconnected generalized learning automata in which the control is transferred from one learning automaton to another[15]. Each state in MDP has a learning automaton that tries to learn the optimal probability distribution of actions during the process. Agents move on this network and in each state they get help from the generalized learning automaton assigned to that state to move to the next state. This is done by using the probability vector of the corresponding earning automaton. In this model, only one learning automaton is active in each time and the transition from one state to another will activates the learning automaton of that state. The activated generalized learning automaton LA_i in the state i will not be informed about the immediate reward yielded from its action a_i in transition from state i to state j. Instead, when state i is visited again, LA_i receives two parts of data: the cumulative reward from the beginning of the process up to the current time step and current global time. Using these data, GLA_i calculates the reward received since the last visit of state i and the corresponding elapsed global time. Then, the input to LA_i is calculated using the following equation:

$$\beta^i(t_i+1) = \frac{\rho^i(t_i+1)}{\eta^i(t_i+1)} \qquad (6)$$

Where $\rho^i(t_i+1)$ is the cumulative total reward generated for action a_i in state i and $\eta^i(t_i+1)$ is the cumulative total elapsed time. This process continues until all probability vectors converge or a pre-specified condition is met. When the number of agents increase and the model extends to multi agent case, more than one learning automaton should be active simultaneously because the states depend on the problem and the environment and the agents could be in different states.

4. The Proposed Algorithm

Here a new algorithm for solving problems in the environment with the goal for learning automata based multi agent system (MAS) is proposed. In this algorithm, the Markov environment as a directed graph is mapped. This mapping is done as a node of the graph indicate the states and directed edges between nodes represent the actions that lead to the transfer agent of the state to another. Each node of the graph, the GLA will be associated with a condition that is on that node mapping. An automaton procedure according to the number of neigh boring nodes is determined. At first, all of the learning automata in all states choose their actions with equal probabilities. It is assumed that the final state of the environment that has the most immediate rewards, as the target. The agents start from the initial situation to achieve the goal, in parallel on the corresponding graph is moving. Each agent continues moving on the graph until it reaches the goal state. After reaching the goal, the path π_j for each agent j ($L\pi_j$), is computed based on the length of the traversed path π_j divided by the reward of reaching the goal state, (R_G), the cost of the path is compared with a threshold value, such as T_j. Depending on the result of comparing all the learning automata along the way receive reward or penalties. This process is repeated for each agent if the path converges to the optimal path or a predetermined criterion is satisfied. For example, assume that the path π_j for agent j is traversed k times. The threshold is defined as the average cost varies according to the following equation. This algorithm is called AlgorithmG1 seen in Figure3.

$$T_j = T_j + (L\pi_j - T_j) / k \qquad (7)$$

Algorithm G1 suffers from the possibility of existence of cycles in the paths taken by agents. This possibility increases with increase in learning rate or reward. Generalized Learning Automata is including a fixed number of action, but in some applications need to Learning automata with the number of variables. Selection is done by an external agent and randomly. To evaluate the performance of network generalized learning automata to control the Markov chain several experiments have been conducted. The environment of experiment is a Grid-world game that includes a 3×3 grid with a goal cell. This environment has also been used previously in [12, 13].Two agents start from the two bottom corners (location 1 and 3) and try, with the least possible number of moves, to find the goal square, which is the top centre. After observing the current state, agents choose their actions simultaneously. They then observe the new location, both agents' rewards, and the action taken by the other agent. In the new location, agents repeat the process above. When at least one agent moves into its goal position, the game restarts. In the new episode, each agent is randomly assigned a new position (except its goal cell). If both agents try to move to the same cell both receive -1. If agents move to two different non-goal cells, both receive zero rewards and if one reaches the goal position, it receives 100 units of reward.

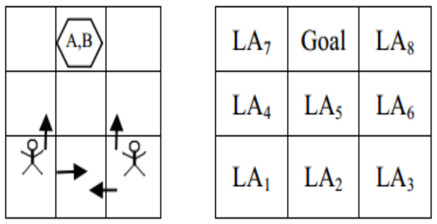

Fig. 2. Grid-world game and learning automata model[5].

ALGORITHM G1

```
1:  Create one Generalized learning automaton for each
    state s and define the set of actions based on equation (5).
2:  Place all agents in starting state;
3:  for every agent j do
4:  set threshold , $k_j$
    //$k_j$ number of the path traversed by agent j from
    starting state to goal state//
5:  end for
6:  for every agent j do in parallel
7:  Repeat
    8:  Activate Generalized learning automat onto
        state s in which the agent j is residing and
        determine the next state m to which the agent
        Moves.
    9:  if state m is the goal state then
        10: Compute the value of path $\pi_j$ taken by agent j
            $L_j(\pi_j)$, to be $L_j(\pi_j)$ /RG
            //$R_G$ is the reward of reaching the goal
            state, $t_j(\pi_j)$ is the length of the traversed path$\pi_j$//
            //Compute the cost of path $\pi j$ taken by agent j//
        11: if $Lj(\pi_j<T_j)$ then
        12: Give reward with $\beta(k) = L_j(\pi_j)$
        13: else
        14: Give penalty with $\beta(k) = 1- L_j(\pi_j)$ ;
        15: end if
        16: $T_j = T_j + (L_j(\pi_j)- T_j )/ k_j$;
            // $T_j$ is the average cost of traversed path//
        17: $k_j=k_j+1$
    18: end if
19: Until ( the probability of path $\pi j$ taken by agent j
    exceeds a pre-specified threshold or a fixed number of
    iterations are passed)
20: end for
```

Fig 3. Algorithm G1.

5. Experimental Results and Discussions

In order to evaluate the performance of the proposed algorithm, several experiments have been conducted. The environment of experiment is a Grid world game that includes a 3×3 grid with a goal cell. Two agents start from the two bottom corners (location 1 and 3) and try, with the least possible number of moves, to find the goal square, which is the top center. After observing the current state, agents choose their actions simultaneously. They then observe the new location, both agents' rewards, and the action taken by the other agent. In the new location, agents repeat the process above. When at least one agent moves into its goal position, the game restarts. In the new episode, each agent is randomly assigned a new position (except its goal cell).

To convert the problem to an MMDP, we consider each cell as a state and the allowable transitions to adjacent cells as actions. In this MMDP the objective is that both agents reach the goal state using distinct paths. If both agents try to move to the same cell, both of their moves will fail and both receive 0.01 units of punishment and stay in their previous position. If agents move to two different non goal cells, both receive zero rewards and if one reaches the goal position, it receives 1 units of reward. The reward function $r_i(t)$for each agent i is based on Equation 8, i=1, 2.

$$ri(t) = \begin{cases} 1 & if\,agent\,i\,reaches\,the\,goal\,state \\ 0.01 & if\,agents\,reached\,different\,non\,goal\,state \\ 0 & if\,both\,agents\,reached\,the\,same\,non\,goal\,state \end{cases} \tag{8}$$

A path in this game represents a sequence of actions from the start to the goal position. In game terminology, such path is called a policy or strategy. The shortest path not interfering the path taken by the other agent is called the optimal path.

Experiment 1: This experiment is conducted to study the impact of learning parameter λ on the probability of optimal path in Algorithm G1. The probability of optimal path is defined as the product of probabilities of the selection of the edges along the optimal path. The plots of average probability of optimal path over the converged runs out of 100 runs are given in Figure 4. For example, It can be seen that in terms of accuracy the best result is obtained when λ= 0. 1.

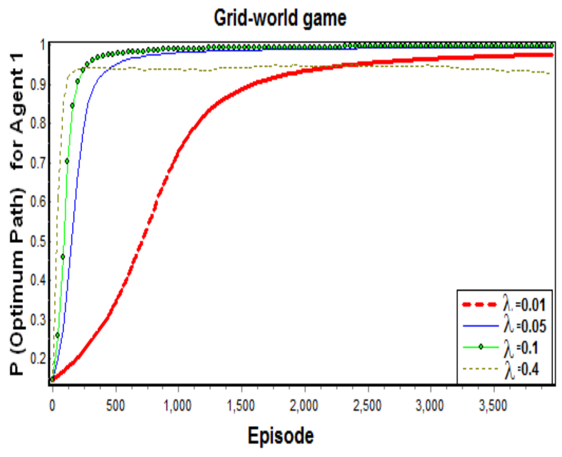

Fig. 4. Probability of optimal path for agent 1 for Algorithm G1 for different learning rate. k=0/01, L=1 , T=0/1.

Experiment2: This experiment is conducted to study the impact of learning parameter λ on the amount of reward received by agent 1 during an episode in Algorithm G1. For this purpose, we plot the reward received by agent 1 per episode for different values of learning parameter λ: 0.01, 0.05, 0.1 and 0.4. Each value reported in this experiment is obtained by averaging over 100runs. We assume that at the beginning of an episode ,each agent starts from starting cell. Figure 5 illustrates the results of this experiment. The result illustrates clearly that choice of value for parameter λ has a large effect on the performance of the proposed algorithm. The results indicate that the best result is obtained when λ= 0.1.

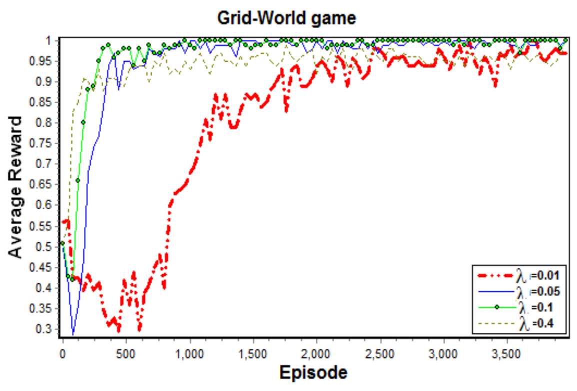

Fig. 5. The impact of learning rate on the reward received by agent1.k=0/01, L=1 , T=0/1.

Experiment 3: This experiment is conducted to study the impact of learning parameter λ of generalized learning algorithm on the number of agent collisions during an episode in Algorithm G1. For this purpose, we plot the number of collisions between agents per episode for different values of learning parameter a: 0.01, 0.05, 0.1 and 0.4. We assume that at the beginning of an episode, each agent starts from starting cell. Each value reported in this experiment is obtained by averaging over 100 runs. Figure 6 summarizes the results of this experiment and shows the results indicate that the best result is obtained when λ= 0.1.

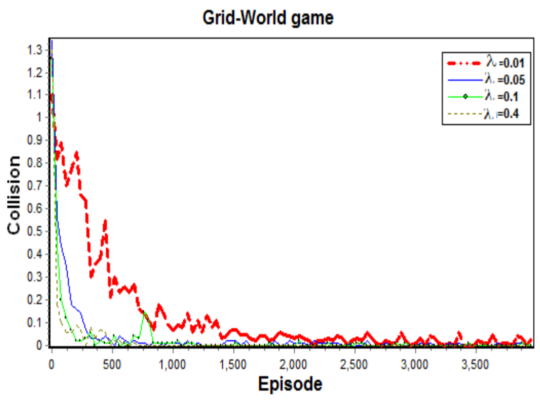

Fig. 6. The impact of learning rate on the number of agent collisions during an episode.k=0/01, L=1 , T=0/1.

Experiment4: In this experiment we compare Algorithm G1 with Algorithm1 [5] in terms of the reward received by agent 1and the number of agent collisions made during an episode Learning parameter $\lambda$for both algorithms is set to 0.1. Each value reported is the average over 100 runs. It is assumed that at the beginning of an episode, each agent starts from starting cell. Figure7 shows the result of experiment. As it is seen Algorithm1 with GLA outperforms Algorithm1 with LA in terms of the number of agent collisions and the average reward received during an episode and Probability of optimal path for agent 1.

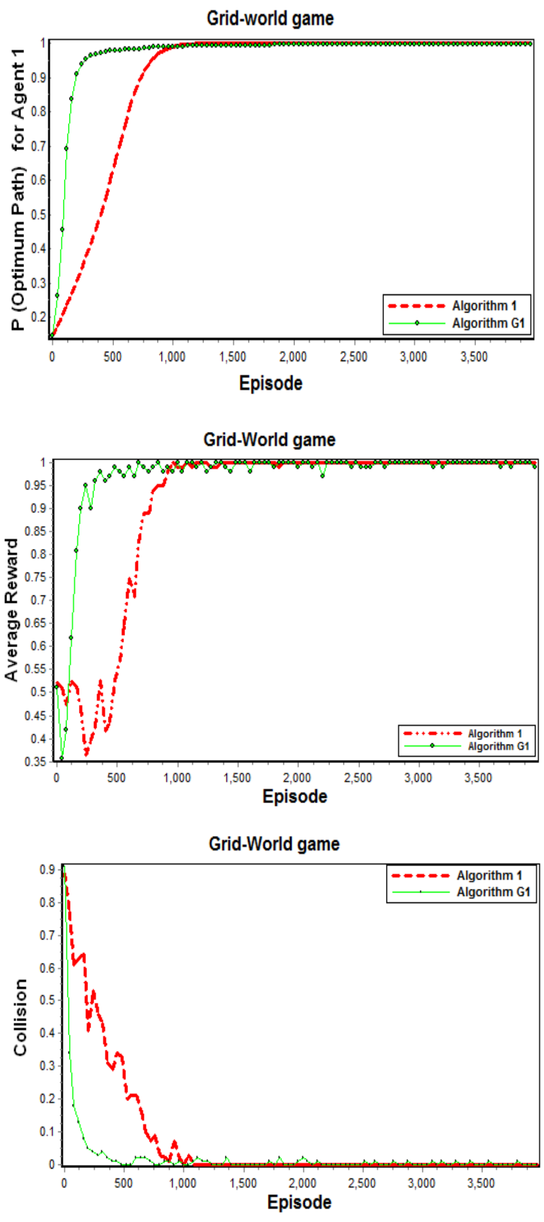

Fig. 7. Comparison of Algorithm G1 with Algorithm1 [5] in terms ofa) average rewardreceivedb) the number of agent collisions c)Probability of optimal path for agent 1.k=0/01, L=1 , T=0/1, λ=0/ 1.

Experiment 5: In this experiment, we investigate the performance of the proposed algorithm and algorithm ILA [14] and 2[5]. In Grid world game state transitions are deterministic, which means the current state and agent's joint action will uniquely determine the next state. In this experiment, the optimal path probability and the reward received by agent 1 per episode have been observed. Table 1 gives the number of iterations required by each algorithm in order the probability of optimal path exceeds 0.97.Figure 8 shows the results compared with the proposed algorithm. As it is seen algorithm G1 outperforms all the other algorithms. Notice that algorithm ILA performs the worst.

Table 1

Maximum number of iterations for the first agent to reachthe optimal path.

Used Algorithm	*Maximum Iterations* *λ=0/1*
Algorithm G1	*1000*
Algorithm 2	*2500*
Algorithm ILA	*4000*

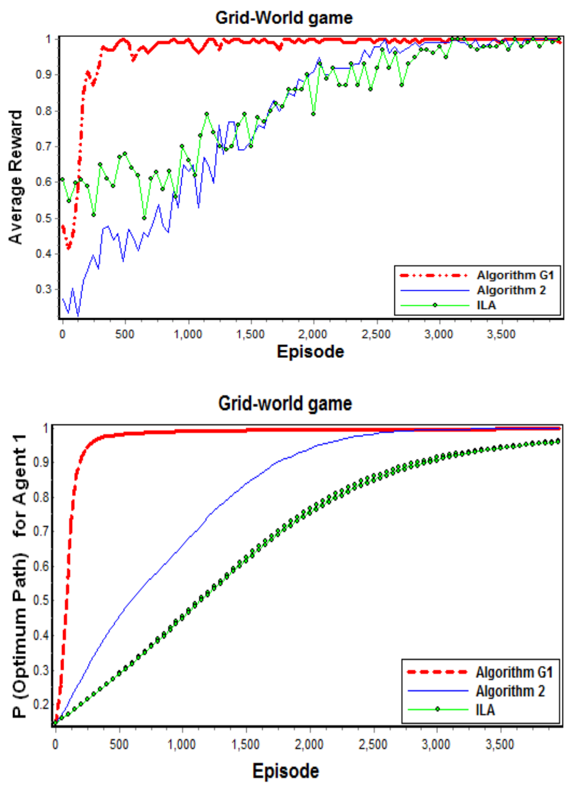

Fig. 8. Comparison of the Algorithms G1 with Algorithms 2[5] and AlgorithmILA[14] in terms of a) average rewardreceived,b) optimal path for agent 1,k=0/01, L=1, T=0/1, λ=0/1.

6. Conclusion

In this paper a new multi agent reinforcement learning algorithm based on generalized learning Automata for finding optimal policies in Markov games was proposed. In the proposed algorithm, the environment of Markov problem is modeled as a directed graph. The nodes of this graph represent the states and directed edges between nodes represent the actions that result in transition from one state to another. Each node of the graph is equipped with a generalized learning automaton whose actions are the outgoing edges of the corresponding node. The agents move on this graph and in each state, they get help from corresponding generalized learning automaton to move to the next state. The proposed multi agent systems were evaluated by applying them to an example of a MMDP called Grid Game. Simulation results showed that the choice of the learning rate has a great impact on the performance of all the proposed algorithms. The results of experimentations showed that the proposed generalized learning automata outperforms the previous learning automata based multi agent system in terms of cost and speed of convergence. In the proposed algorithm there are limitations, such as requiring that the problem involves reaching a goal state. As future work can be examined to improve the efficiency of multi agent systems by generalized learning automata that agents are better able to adapt to changing circumstances and unforeseen events.

References

[1] A. Nowé, K.Verbeeck,M.Peeters,"Learning Automataas a Basis for Multi Agent Reinforcement Learning,"in Computer Science. vol. 3898: Springer, Berlin,2006, pp 71-85

[2] P. Vrancx, “Decentralised Rein forcement Learning in Markov Games," for the degree Doctor of philosophy in Sciences, Computational Modeling Lab Department of Computer Science Faculty of Sciences Vrije Universiteit Brussel, 2010, pp. 33.

[3] B. masoumi, M. R. Meybodi, "Speeding up learning automata based multi agent systems using the concepts of stigmergy and entropy,"in Expert Systems with Applications, 8105–8118,2011.

[4] L. Busniu, R. Babuska, and B. Schutter, "A Comprehensive Survey of Multiagent Reinforcement Learning," IEEE Transaction on System, Man, Cybern, vol. 38, pp. 156-171, 2008.

[5] M. Song, J. Bai, and R. Chen, "A New Learning Algorithm for Cooperative Agents in General-Sum Games," in the Sixth International Conference on Machine Learning and Cybernetics Hong Kong, 2007, pp. 50-54.

[6] P. Vrancx, K. Verbeeck, and A. Nowé, "Generalizedlearning automata for Multi-agent Reinforcement Learning", AI Communications, 2010.

[7] K.Verbeeck, P. Vrancx, and A. Nowé, "Using Generalized Learning Automata for State Space Aggregation in MAS," Computational Modeling Lab, 2010.

[8] B.Masoumi, M. R. Meybodi and F. Abtahi, “Learning Automata based Algorithms for Finding Optimal Policies in Fully Cooperative Markov Games" in Conf. Przeglad Elektrotechniczny, 2012.

[9] M. Thathachar and P. Sastry, "Networks of Learning Automata for Online Stochastic Optimization," Kluwer Academic Publishers, 2004.

[10] P. Vrancx, K. Verbeeck, and A. Nowé, "Generalized learning automata for Multi-agent Reinforcement Learning", AI Communications, 2010.

[11] R. Williams, “Simple statistical gradient- following algorithms for connectionist reinforcement learning,” Journal of MachineLearning, vol. 8, no. 3, pp. 229–256, 1992.

[12] B. masoumi, M. R. Meybodi, " Learning automata based multi-agent system algorithm for finding optimal policies in markov games " Asian Journal of Control, Vol. 14, No. 4, pp. 116, July 2012 .

[13] H. Qio, F. Szidarovszky, Rozenblit, and L. Yong, " Multi-agent Learning Model with Bargaining," in the 38th Conference on Winter Simulation Monterey, California, 2006, pp. 934 - 940.

[14] A. Nowe, K. Verbeeck, Colonies of Learning Automata, IEEE Transactions on Systems, Man and Cybernetics, 32 (2002) ,pp 772-780.

[15] R. M. Wheeler and K. S. Narendra, "Decentralized Learning in Finite Markov Chains," IEEE Transactions on Automatic Control, vol. 31, pp. 519-526, 1986.