Multi-agent Learning in Extensive Games with Complete Information *

Pu Huang
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
phuang@cs.cmu.edu

Katia Sycara
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
katia@cs.cmu.edu

ABSTRACT

Learning in a multi-agent system is difficult because the learning environment jointly created by all learning agents is *time-variant*. This paper studies the model of multi-agent learning in complete-information extensive games (CEGs). We provide two provably convergent algorithms for this model. Both algorithms utilize the special structure of CEGs and guarantee both individual and collective convergence. Our work contributes to the multi-agent learning literature in several aspects: 1. We identify a model of multi-agent learning, namely, learning in CEGs, and provide two provably convergent algorithms for this model. 2. We explicitly address the environment-shifting problem and show that how patient agents can collectively learn to play equilibrium strategies. 3. Many game-theoretical work on learning uses a technique called *fictitious play*, which requires agents to build beliefs about their opponents. For our model of learning in CEGs, we show it is true that agents can collectively converge to the sub-game perfect equilibrium (SPE) by repeatedly reinforcing their previous success/failure experience; no belief building is necessary.

Categories and Subject Descriptors: [**ARTIFICIAL INTELLIGENCE**]: I.2.11, Distributed Artificial Intelligence - Multiagent systems; I.2.6 Learning - Concept learning. General Terms: Algorithms, Economics.

Keywords

Multi-agent systems, Learning, Extensive games.

1. INTRODUCTION

Learning in general refers to the process of acquiring and applying knowledge to improve the behavior and performance in a certain task. Machine learning techniques for building intelligent agents that can learn automatically has drawn much attention from computer scientists in the last decade. Most work in machine learning to date has dealt with single-agent learning problems. In these problems, a single agent operates in a stochastic environment that evolves according to a Markov process; the agent must decide which action to take in every environment state and the decision made stochastically determines the successive state the agent will reach as well as the immediate reward the agent will get. The environment is assumed *time-invariant*, i.e., the parameters characterizing the Markov process do not change over time. The agent's goal is to find an optimal policy (i.e., decide which action to take in every environment state) that maximizes the infinite-horizon discounted reward or the long-run average reward through learning. These types of problems are well modeled as Markov Decision Processes (MDPs). The structure of the optimal policy of MDPs has been well understood [1]; many learning algorithms have been proposed [15]; many applications have been developed [8].

In contrast to the well-developed understanding of single-agent learning, the field of multi-agent learning is still far from mature. A multi-agent system consists of a collection of *autonomous* and *self-interested* agents that interact with each other. Multi-agent systems differentiate themselves from single-agent systems not only by the number of agents, but also by the fact that every agent is autonomous. Autonomy means every agent is an independent entity, and there exists no centralized control for the system as a whole. A system with multiple agents and a centralized control still falls into the single-agent category, because the centralized control mechanism can be used to "force" the agents to pursue a single goal (for example, to maximize the collective welfare of all the agents). Agents in a multi-agent system are self-interested in the sense that they have, and are free to pursue, their own interests, which may or may not align with each other.

The presence of multiple autonomous and self-interested agents makes multi-agent learning a difficult problem. The reason is that if every agent is learning, then everyone faces a *time-variant* environment. Since all agents are learning, how the other learning agents will react to the strategy improvement of an individual agent will depend on how this agent improves its own strategy, which, in turn, determines how the learning environment of this agent will shift. (To an

*This research was supported by CoABS DARPA contract F30602-98-2-0138 and AFOSR contract F49620-01-1-0542.

individual agent, all the other agents are a part of its environment.) Therefore, the way in which a single agent learns affects how its environment will shift over time. To further complicate the matter, different agents may respond to the policy improvement of the same agent differently, which makes it even harder for the agents to anticipate how the environment will shift.

This paper examines a specific multi-agent learning model: learning in complete-information extensive games (CEGs). Various negotiation processes can be modeled as CEGs (see, e.g. [4]). As e-commerce prevails, we expect more intelligent agents will be deployed as human delegates to play market games. In order to be "intelligent", agents must acquire the ability to learn. In this paper, we will study general multi-agent learning algorithms for CEGs without mentioning detailed models for on-line agent negotiation, which is out of the scope this work.

An extensive game has a tree-like structure that defines the detailed sequential moves of participating agents. Complete information means that every agent can fully observe the actions taken by all other agents (see, e.g. [4]). Because of the tree-like structure and the presence of complete information, a CEG admits a strong solution concept: the subgame perfect equilibrium (SPE) (see Section 2 for details). In this paper, we propose two reinforcement-learning algorithms for multi-agent learning in CEGs. We also prove that both algorithms are convergent to the SPE. Our results are based on the observation that multi-agent learning in CEGs is closely related to single-agent learning in MDPs. Actually, if only one agent is learning and all others behave according to some fixed strategies, the CEG model degenerates to an MDP (see Section 3 for details), and the learning agent faces the problem of maximizing its long-run average reward. The MDPs derived for CEGs inherit the tree-like structure, and because of this structure, we can slightly change the well-known "learning automaton" algorithm [13], and apply the modified version for multi-agent learning in CEGs. We call this algorithm Multi-agent Learning Automata (MLA). If all agents are patient enough (use small learning rates) and apply MLA simultaneously, we show that the collective learning process actually converges to the SPE. MLA reinforces actions by estimating the "value" of each action; the values themselves are not recorded. If we keep the values recorded, update them in the learning process, and then choose actions according these values, we get another provably convergent multi-agent learning algorithm for CEGs. We call it MQ-learning.

Our work contributes to the multi-agent learning literature in several aspects: 1. We identify a model of multi-agent learning, namely, learning in CEGs, and provide two provably convergent algorithms for this model. 2. We explicitly address the environment-shifting problem and show that how patient agents can collectively learn to play equilibrium strategies. 3. Many game-theoretical work on learning uses a technique called *fictitious play*, which requires agents to build beliefs about their opponents. For the particular model of learning in CEGs, we show it is actually true that agents can collectively converge to the SPE (a fully rational outcome) by repeatedly reinforcing their previous success/failure experience; no belief building is necessary.

The rest of the paper is organized as follows. Section 2 introduces CEGs and the concept of SPE. Section 3 discusses the relation between multi-agent learning in CEGs and single-agent learning in MDPs. In Section 4 and 5 we present the MLA algorithm and the MQ-learning algorithm and show that both converge to the SPE. Section 6 numerically tests our two algorithms. Relate work is discussed in Section 7. Section 8 concludes this paper.

2. EXTENSIVE GAMES WITH COMPLETE INFORMATION

An extensive game played by I players can be represented by a tree G. Every non-leaf node s of G is owned by a player $i \in I$. A player owning a non-leaf node s can take any action $a \in A(s)$ when it is his turn to play, where $A(s)$ is the set of all available actions in node s. After this player takes an action a, the game moves to a successive node $s' =< s, a >$. Depending on whether s' is a non-leaf node or a leaf node, the game continues or stops: if s' is a non-leaf node, the game progresses and the player who owns node s' will take his turn to move; otherwise, if s' is a leaf node, the game ends and every player i gets a reward (payoff) $r_i(s')$. Rewards in a leaf node s' are represented by a vector $\overrightarrow{r(s')}$, and the i's component $r_i(s')$ represents player i's reward.

In every non-leaf node s, player i follows a strategy (policy) $\pi_i(s)$ to play the game. Player i's strategy $\pi_i(s)$ in node s is a probability distribution over $A(s)$. In other words, when the game progresses to a node s owned by player i, he selects an action $a \in A(s)$ to play according to the probability distribution $\pi_i(s)$. All the $\pi_i(s)$'s in all the nodes owned by player i constitute player i's strategy π_i for the whole game. All the π_i's of all the players constitute a strategy profile $\pi = \{\pi_1, \pi_2, ..., \pi_N\}$. A player i's strategy π_i is a contingency plan that specifies which action to take in every node owned by him. All these contingency plans of all the players form a strategy profile of the game. Once the structure of the game tree G and the strategy profile π of all the players are given, any individual player i's *expected* reward, $R_i(\pi_i|\pi_{-i}, G)$, is determined as well. We can calculate $R_i(\pi_i|\pi_{-i}, G)$ as

$$R_i(\pi_i|\pi_{-i}, G) = \sum_{s \in LF(G)} P^{\pi}(s) r_i(s),$$

where $LF(G)$ is the set of all leaf nodes in the game tree G; $P^{\pi}(s)$ is the probability that the game ends at a leaf node s, given that the strategy profile of all the players is π; and π_{-i} denotes the strategy profile of all the other players excluding player i.

An extensive game is said to be with complete information if all the actions taken previously are observable to every player; otherwise, the game is with incomplete information. The above definition of strategy is valid for complete information games only. For incomplete information games, because of unobservable information, a player's strategy is contingent on *information sets* instead of single nodes (see, e.g. [4]). This paper deals with learning in complete-information extensive games (CEGs) only. We'll discuss issues related to incomplete-information games in the conclusion section.

Figure 1 is a CEG played by three players. In this game, non-leaf nodes are labeled with a pair representing the name

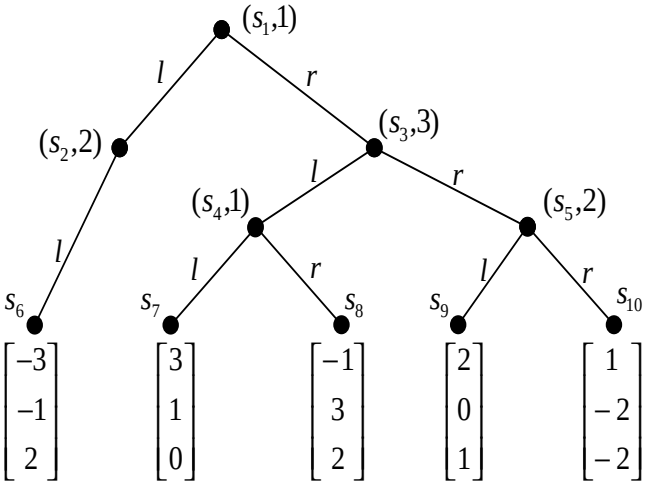

Figure 1: An extensive game played by three players. Non-leaf nodes are labeled with a pair representing the name and the owner; leaf nodes are labeled with the name only. The payoffs of the players are shown as vectors below the leaf nodes.

and the owner; leaf nodes are labeled with the name only. For example, label $(s_1, 1)$ of the root node represents that this node is named s_1 and owned by player 1; label s_6 is the name of a leaf node that is the direct successor of node $(s_2, 2)$. Two actions, we call them l and r, are available in each non-leaf node except s_2, where only action l is available. The payoffs are shown in the figure as vectors below the leaf nodes. The first entry in every vector is the payoff of player 1, the second entry is the payoff of player 2, and so on.

An important solution concept of CEGs is sub-game perfect equilibrium (SPE). A sub-game of an extensive game is the game represented by a sub-tree G_s, rooted at a node s, of the original game tree G. A strategy profile $\pi^* = \{\pi_i^*, \pi_{-i}^*\}$ is a SPE if for every sub-game $G_s \subseteq G$ and every player i, strategy π_i^* of player i maximizes his expected reward, i.e.,

$$R_i(\pi_i^* | \pi_{-i}^*, G_s) \geq R_i(\pi_i | \pi_{-i}^*, G_s)$$

for any strategy π_i other than π_i^*. An equilibrium strategy profile is a stable profile in the sense that any individual player i will not deviate from his strategy π_i^* if all the other players stick to their equilibrium strategy π_{-i}^*, which, in turn, holds because no player deviates.

If the reward vector in every leaf node is known to every player, backward induction can be used to find the SPE. Backward induction finds the SPE of a CEG by propagating the reward vectors in the leaf nodes to the non-leaf nodes, starting from the leaf nodes, ending at the root. The reward vector $\overrightarrow{r(s)}$ of a non-leaf node s represents the rewards all the players would obtain by playing the SPE strategies in the sub-game G_s rooted at node s. The procedure of backward induction is described as follows.

Let $B(1)$ denote the set of nodes whose successors are leaf nodes only, i.e.,

$$B(1) = \{s | < s, a > \in LF(G), \text{ for } \forall a \in A(s)\}. \quad (1)$$

Then in every node $s \in B(1)$, the owner of s (suppose it to be player i) would choose an action a that leads to a leaf node $< s, a >$ where he can get the maximum reward, i.e., player i would choose an action a such that

$$a = \arg\max_{b \in A(s)} \{r_i(< s, b >) | s \in B(1)\}.$$

The reward player i gets in node s by choosing action a is

$$r_i(s) = r_i(< s, a >), \quad (2)$$

and every other player $i' \neq i$ gets reward

$$r_{i'}(s) = r_{i'}(< s, a >). \quad (3)$$

Putting equations (2) and (3) together, the reward vector $\overrightarrow{r(s)}$ in node s can be represented by

$$\overrightarrow{r(s)} = \overrightarrow{r(< s, a >)}, \text{ for } \forall s \in B(1). \quad (4)$$

We call $B(1)$ level-1 set. Similarly, we can define level-2 set $B(2)$ as the set of nodes whose successors are leaf nodes *or* nodes in $B(1)$, i.e.,

$$\begin{aligned} B(2) \quad = \quad & \{s | < s, a > \in LF(G) \cup B(1) \text{ for } \forall a \in A(s) \\ & \text{and } s \notin B(1)\}. \end{aligned} \quad (5)$$

For every node $s \in B(2)$, the owner of s (suppose to be player j) would choose an action a such that

$$a = \arg\max_{b \in A(s)} \{r_j(< s, b >) | s \in B(2)\},$$

and therefore the reward vector in node s is determined by

$$\overrightarrow{r(s)} = \overrightarrow{r(< s, a >)}, \text{ for } \forall s \in B(2). \quad (6)$$

In general, level k set $B(k)$ is defined as

$$\begin{aligned} B(k) \quad = & \{s | < s, a > \in LF(G) \cup B(1) \cup ... \cup B(k-1) \\ & \text{for } \forall a \in A(s) \text{ and } s \notin B(1) \cup ... \cup B(k-1)\}. \end{aligned} \quad (7)$$

For every node $s \in B(k)$, the owner of s would choose an action a that maximizes his own reward and the reward vector in node s is thus determined by

$$\overrightarrow{r(s)} = \overrightarrow{r(< s, a >)}, \text{ for } \forall s \in B(k). \quad (8)$$

Continue this procedure until reaching the root of the game tree. The reward vector in the root then is the SPE reward, and every player's strategy in every node constitutes the SPE strategy profile.

In this paper, we assume for every play i and every pair of distinct terminal nodes s and s', $r_i(s) \neq r_i(s')$, i.e., there is no tie for every player. This type of CEG is called *generic* and has a unique SPE. In the game shown in Figure 1, the unique SPE reward is the reward vector in the leaf node s_9.

3. LEARNING IN EXTENSIVE GAMES

If only one player in a CEG is learning, then the environment this single learning player faces is an MDP. To give a concrete example, refer to Figure 1. Suppose both players 2 and 3 have chosen to move according to a fixed strategy. Player 2's strategy is: $\pi_2(s_2) = \{1\}$, and $\pi_2(s_5) = \{0.9, 0.1\}$, i.e., player 2 always chooses action l in state s_2, and selects action l and r in state s_5 with probability 0.9 and 0.1 respectively. Player 3's strategy is $\pi_3(s_3) = \{0.4, 0.6\}$. Then the MDP the single learner, player 1, faces is shown in Figure

2. The states of the MDP consist of all the non-leaf nodes owned by player 1 and all the leaf nodes. The actions available in every state are l and r, represented in the figure by solid-line arrows. Taking certain actions in some states leads to another state with probability one. We mark these actions with two numbers, the first one is the probability, and the second one is the payoff. For instance, if player 1 chooses action r in node s_4, he will move to s_8 with probability 1 and get payoff -1. Taking other actions probabilistically leads to several other states. For instance, if player 1 takes action r in state s_1, then he will move to state s_4, s_9, and s_{10} with probabilities $p_1 = 0.4$, $p_2 = 0.54$, and $p_3 = 0.06$ respectively, getting payoffs 0, 2, and 1 respectively. We use dashed-line arrows to represent the transition from one state to several other states. Each dashed-line arrow is marked with the probability of this transition and the payoff obtained by player 1.

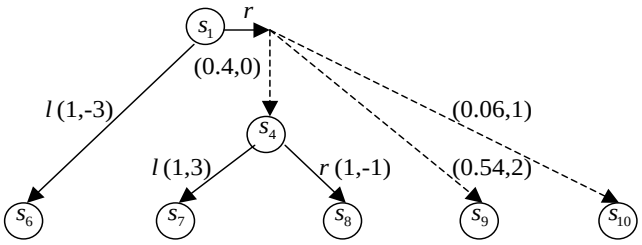

Figure 2: The tree-like MDP player 1 faces assuming he is the only learner and all other players move according to some fixed strategies.

If the single learner in a CEG does not own the root of the game, such as player 2 in Figure 1, then the learning problem of this learner consists of several tree-like MDPs. For example, if players 1 and 3 have fixed strategies and player 2 is learning, then player 2 learns on two separate tree-like MDPs, one consists of states s_2 and s_6, the other consists of states s_5, s_9, and s_{10}. Learning on two, or more, MDPs poses no extra difficulty here. The learner updates his strategy in an MDP only when some states in this MDP are visited in a round of game play; otherwise, if no state is visited in a play, the learner just keeps his previous strategy.

Players in a CEG learn through repeatedly playing the same game. Every player has his own goal of maximizing his own long-run average reward. We call a round of game play an episode and index episodes by t. Given an extensive game G, if player i is the only learner (thus only his strategy π_i^t changes over time), and all the other players move according to a fixed strategy π_{-i}, then player i's average reward in N episodes is defined as

$$AR_i^N(\pi_i^t) = \frac{\sum_{t=1}^{N} R_i(\pi_i^t|\pi_{-i}, G)}{N},$$

where $R_i(\pi_i^t|\pi_{-i}, G)$ is the expected payoff player i obtains in episode t. The long-run average reward of player i is defined as the limit of $AR_i^N(\pi_i^t)$ as N goes to infinity, i.e.,

$$AR_i(\pi_i^t) = \lim_{N\to\infty} AR_i^N(\pi_i^t).$$

For any fixed strategy π_{-i} of player i, there exists a strategy π_i^* for this player such that π_i^* maximizes his expected reward $R_i(\pi_i^*|\pi_{-i}, G)$ in one round of game play. If, through learning, player i's long-run average reward $AR_i(\pi_i^t)$ converges to $R_i(\pi_i^*|\pi_{-i}, G)$, we say the learning algorithm adopted by player i is individually convergent.

If all players in an extensive game G are learning, then every player's strategy changes over time. Let $\pi^t = \{\pi_i^t, \pi_{-i}^t\}$ denote the strategy profile of all the players in episode t, again, define player i's average reward in N episodes and long-run average reward as

$$AR_i^N(\pi_i^t|\pi_{-i}^t) = \frac{\sum_{t=1}^{N} R_i(\pi_i^t|\pi_{-i}^t, G)}{N},$$

$$AR_i(\pi_i^t|\pi_{-i}^t) = \lim_{N\to\infty} AR_i^N(\pi_i^t|\pi_{-i}^t).$$

Let $\pi^* = \{\pi_i^*, \pi_{-i}^*\}$ denote the SPE strategy profile of game G. Then if every player i's long-run average reward converges to his SPE reward $R_i(\pi_i^*|\pi_{-i}^*, G)$, we say the learning algorithm adopted by the players is *collectively convergent*. The distinction between individual convergence and collective convergence is that the former assumes a time-invariant environment, while the later does not.

Remarks: Note that rewards are averaged over episodes and every round of game play causes the episode index increases one. This is different from the definition of average-reward in MDPs, in which time index increases every time when a state transition happens.

One commonly asked question is why a player, say, player 2 in Figure 1, would follow some mixed strategy, say, in node s_5, instead of sticking to the best strategy in this node, choosing action l? There are two reasons: (1) In the learning process, playing greedily is not necessarily of a player's best interest. For example, if player 1 always takes action l in node s_4, player 3 will eventually "learn" this fact and will never take action l in node s_3, and consequently player 1 will have no chance to get the best reward it would get in leaf node s_7. Therefore, players randomize their strategies to explore the best rewards they *can* get. (2) We assume players only know their own payoffs in leaf nodes. (Both of our algorithms do not require a player to know its opponents' payoffs.) Without knowing other players' payoffs, a player can not decide taking which action (pure strategy) is of its best interest and therefore need to randomize its strategy to explore the best reward it *can* get. This latter assumption is critical for modeling negotiations. In a negotiation process, it is usual that agents can observe their opponents' moves, but not their final rewards/payoffs.

4. MULTI-AGENT LEARNING AUTOMATA (MLA)

The learning automaton algorithm has been shown convergent for *single-agent* average-reward learning in *ergodic* [1] MDPs [11, 13]. Because of the special tree-like structure of the MDPs derived from CEGs, we can slightly change the standard learning automaton algorithm and prove that

[1] An MDP is ergodic if *every policy* results in a single recurrent class, though different policies may result in different recurrent sets. See [11] for details.

the modified version is actually both individually and collectively convergent. We call the modified version Multi-agent Learning Automata (MLA). We state MLA for multi-agent learning in CEGs, the single-agent version for learning in tree-like MDPs is the same. The algorithm is as follows:

In every non-leaf node s of the game, the owner of this node (assume to be player i) keeps a probability distribution $\pi_i^t(s)$ over all actions available in this node, i.e., $\pi_i^t(s) = \{p_i^t(a)\}$, where $a \in A(s)$ and $\sum_{a \in A(s)} p_i^t(a) = 1$. Every time when it is player i's turn to move, he takes an action according to this probability distribution. Assume player i takes an action a in episode t, then at the end of this episode, the probability distribution $\pi_i^t(s)$ is updated according to

$$\begin{aligned} p_i^{t+1}(a) &= p_i^t(a) + \alpha\beta(1 - p_i^t(a)), \\ p_i^{t+1}(b) &= p_i^t(b) - \alpha\beta p_i^t(b), \qquad \forall b \neq a, \end{aligned} \tag{9}$$

where $0 < \alpha < 1$ is a learning rate, and β is the *scaled* reward received in episode t. To ensure that $\pi_i^t(s)$ is a probability distribution at any time, β is scaled to fit into interval [0, 1], with 0 representing the lowest reward the 1 representing the highest reward.

Theorem 1: For any $\varepsilon > 0$, there exists a learning rate $0 < \alpha < 1$ such that if the learning rate of every player is less than α, then the long-run average reward every player gets by following the MLA algorithm shown above will converge to the $\varepsilon-$range of his SPE reward.

Proof: We prove this theorem by induction on the depth of the game tree. Define the level-1 set $B(1)$ as in (1), then in every node $s \in B(1)$, taking an action will lead to an immediate reward. By the convergence of the single-automaton algorithm [13], given any $\varepsilon > 0$, there exists $0 < \alpha(s) < 1$, such that if the owner of the node s (suppose it to be player i) chooses a learning rate less than $\alpha(s)$, then his strategy $\pi_i^t(s)$ will converge close enough to the optimal strategy $\pi_i^*(s)$ (under Euclidean metric) such that $AR_i(\pi_i^t(s))$, the long-run average reward player i gets by following strategy $\pi_i^t(s)$, satisfies

$$|AR_i(\pi_i^t(s)) - r_i(s)| < \varepsilon, \tag{10}$$

where $r_i(s)$ is the maximum reward player i can obtain in node s, as defined in (2). For any other player $i' \neq i$ who does not own node s, $\pi_i^t(s)$ converges to $\pi_i^*(s)$ means that the long-run average reward player i' gets in node s will also converge to the reward she would get if player i follows the optimal strategy $\pi_i^*(s)$, i.e.,

$$|AR_{i'}(\pi_i^t(s)) - r_{i'}(s)| < \varepsilon_{i'}, \tag{11}$$

where $r_{i'}(s)$ is defined in (3). Note that $\varepsilon_{i'}$ in (11) may not equal ε in (10). However, by decreasing $\alpha(s)$, the upper-bound of learning rate in node s, we can always control how close $\pi_i^t(s)$ will converge to $\pi_i^*(s)$ and thus control $\varepsilon_{i'}$; therefore, without loss of generality, the following statement holds:

For any node $s \in B(1)$, given any $\varepsilon > 0$, there exists $0 < \alpha(s) < 1$, such that if the owner of the node s (suppose it to be player i) chooses a learning rate less than $a(s)$, then

$$|\overrightarrow{AR(\pi_i^t(s))} - \overrightarrow{r(s)}| < \varepsilon \overrightarrow{1}, \text{ for } \forall s \in B(1), \tag{12}$$

where $\overrightarrow{1}$ is a vector with all entries equal to one; $\overrightarrow{AR(\pi_i^t(s))} = \{AR_i(\pi_i^t(s)),\ AR_{i'}(\pi_i^t(s))\}$ is a vector representing the average rewards all players will get in node s if player i, the owner of node s, follows strategy $\pi_i^t(s)$; $\overrightarrow{r(s)}$, as defined in (4), is the SPE reward vector every player gets if they play SPE strategy in the sub-game rooted in node s.

Now we move onto the level-2 set $B(2)$. In any node $w \in B(2)$, taking an action leads to either a leaf node or a node in $B(1)$. Without loss of generality, assume all actions lead to a node in $B(1)$. After the average reward vector in every node of $B(1)$ has converged to the $\varepsilon-$range of the SPE reward of this node, as shown in (12), the learning problem in the nodes of $B(2)$ is the same as that in the nodes of $B(1)$, except that we cannot pinpoint the rewards to exact numbers but constraint them into intervals. More specifically, suppose player j owns a node $w \in B(2)$ and takes an action $b \in A(w)$ that leads to a successive node $s =< w, b >\in B(1)$, then because of the convergence of the learning process in node $s \in B(1)$, the reward player j gets by taking action b in node w would fall into the interval $[r_j(s) - \varepsilon,\quad r_j(s) + \varepsilon]$, where $r_j(s)$ is the jth entry of $\overrightarrow{r(s)}$. Therefore, by the convergence of the single-automaton algorithm, there exists an upper bound $0 < \alpha(w) < 1$, such that if player j chooses a learning rate less than $\alpha(w)$, his average reward in node w converges to the SPE reward of the sub-game rooted in node w, i.e.

$$|AR_j(\pi_j^t(w)) - r_j(w)| < 2\varepsilon,$$

where $r_j(w)$ is SPE reward player j gets in the sub-game rooted in node w. Again, since we can always control the convergence range by controlling $\alpha(w)$, we have

$$|\overrightarrow{AR(\pi_j^t(w))} - \overrightarrow{r(w)}| < 2\varepsilon \overrightarrow{1}, \text{ for } \forall w \in B(2).$$

By induction, we conclude that Theorem 1 holds. It is also clear that the upper bound α of the whole game is the minimum of the upper bounds in all nodes. □

Theorem 1 says that the MLA algorithm is collectively convergent for multi-agent learning in CEGs. If only one player is learning and all others follow fixed strategies, then as shown in Section 3, the CEGs degenerates to tree-like MDPs. By using the same induction technique as shown above, we can show that the MLA algorithm is also individually convergent.

Corollary 1: Give any $\varepsilon > 0$, a single learning player in a CEG can use the learning automaton algorithm and choose a small enough learning rate to guarantee that his long-run average reward converges to the $\varepsilon-$range of the maximum reward he can get.

5. MQ-LEARNING

The MLA algorithm uses the reward obtained in every episode to reinforce the strategy in every node of a CEG. If we keep a value for each node-action pair and use the reward obtained in every episode to reinforce these values, we get another provably convergent algorithm, MQ –learning. The name stands for Multi-agent Q-learning, because the idea of reinforcing values is derived from the Q-learning algorithm. Here is the algorithm:

In every non-leaf node s of the game, the owner of this node (let it be player i) keeps a vector $\overrightarrow{Q_i^t(s)}$ storing the q-values of all actions available in this node, i.e. $\overrightarrow{Q_i^t(s)} = \{q_i^t(s,a)\}$, for every $a \in A(s)$. Every time when it is player i's turn to move, he favors the action with the maximum q-value and at the same time uniformly explores other actions. More specifically, player i's strategy in node s is

$$\pi_i^t(s) = \begin{cases} \arg\max_{b \in A(s)}\{q_i^t(s,b)\} & \text{prob. } 1-\sigma \\ \text{uniform}\{d | d \neq \arg\max_{b \in A(s)}\{q_i^t(s,b)\}\} & \text{prob. } \sigma \end{cases},$$

i.e., with probability $1-\sigma$, player i chooses the action with the maximum q-value, and with probability σ, player i chooses other actions uniformly.

Assume player i takes an action a in episode t, and then at the end of this episode, the q-vector in node s is updated according to

$$\begin{aligned} q_i^{t+1}(s,a) &= (1-\gamma) q_i^t(s,a) + \gamma\beta, \\ q_i^{t+1}(s,b) &= q_i^t(s,b), \qquad \forall b \neq a, \end{aligned} \tag{13}$$

where $0 < \gamma < 1$ is a learning rate, and β is the reward received at the end of episode t. Here we do not need to re-scale the reward β.

Theorem 2: For any $\varepsilon > 0$, there exists an exploration threshold $0 < \sigma < 1$, such that if the exploration threshold of every player is less than σ, then for any node s and any $a \in A(s)$, the expected q-value of this pair $E[q_i^t(s,a)]$ converges to the $\varepsilon-$range of the SPE reward of the sub-game rooted in node $< s, a >$.

Proof: Again, we prove this theorem by induction on the depth of the game tree. Take an arbitrary node $s \in B(1)$ and assume the owner of this node is player i; our first observation is that for any $a \in A(s)$, $q_i^t(s,a)$ converges to the reward player i would get if the game ends at the leaf node $< s, a >$. Actually, updating procedure (13) defines a difference equation and the solution of this difference equation is

$$q_i^\tau(s,a) = \beta + (q_i^0(s,a) - \beta)(1-\gamma)^\tau, \tag{14}$$

where $q_i^0(s,a)$ is the initial q-value of action a, τ indexes the number of times action a has been taken[2]. Given that $0 < \gamma < 1$, the above solution is globally stable, i.e., $q_i^\tau(s,a)$ converges to β from any initial point $q_i^0(s,a)$. Note that taking an action in $s \in B(1)$ leads to a leaf node $< s, a >$, and thus the reward β is a deterministic number equal to $r_i(< s, a >)$. Therefore, given $0 < \gamma < 1$, $q_i^\tau(s,a)$ converges to $r_i(< s, a >)$ exponentially with any given initial value.

Now we move on to level-2 set $B(2)$. In any node $w \in B(2)$, taking an action b leads to either a leaf node or a node in $B(1)$. If it is a leaf node, for the same reason stated above, the q-value converges. Otherwise, if $s =< w, a >$ is a non-leaf node belonged to $B(1)$, then the reward player j gets by taking action b in node w is a random variable: if the owner of node s chooses the SPE strategy (with probability $1-\sigma$), the reward is $r_j(s)$, where $r_j(s)$ is the SPE reward player j gets in node s; if the owner of node s does not choose the SPE strategy (with probability σ), then the reward is a random variable u depending on which sub-optimal action has been taken. Put together, player j's reward is $\beta = (1-\sigma) r_j(s) + \sigma u$. Since solution (14) is globally stable, the expected q-value $E[q_j^\tau(w,b)]$ converges to $E[\beta] = (1-\sigma) r_j(s) + \sigma E[u]$. With small enough exploration threshold σ, $E[q_j^\tau(w,b)]$ can be bounded as

$$|E[q_j^\tau(w,a)] - r_j(s)| < \varepsilon, \text{ for } \forall w \in B(2) \text{ and } \forall b \in A(w),$$

where $s =< w, a >$ is a node in $B(1)$.

By induction, we conclude that Theorem 2 holds. The exploration threshold for the whole game is the minimum of the exploration threshold in all nodes. □

There are two methods available for a player j to estimate the expected q-value $E[q_j^\tau(w,b)]$: The first one is to average $q_j^\tau(w,b)$ over *local time* τ (long-run average reward) to approximate $E[q_j^\tau(w,b)]$. Due to the Theorem of Large Numbers, this average converges to the expectation. The second way is to decrease the learning rate γ to suppress the variance of $q_j^\tau(w,b)$. Theoretically, as in Q-learning, if player j's learning rate γ^τ at local time τ satisfies

$$\sum_{\tau=0}^{\infty} \frac{1}{\gamma^\tau} = \infty, \quad \text{and} \quad \sum_{\tau=0}^{\infty} \frac{1}{(\gamma^\tau)^2} < \infty,$$

$q_j^\tau(w,b)$ itself converges to its expectations and player j can directly use $q_j^\tau(w,b)$ to estimate $E[q_j^\tau(w,b)]$.

Again, If only one player is learning and all others follow fixed strategies, then by using the same induction technique, we can show that the MQ-learning algorithm is also individually convergent.

Corollary 2: Give any $\varepsilon > 0$, a single learner in a CEG can use MQ-learning and chooses a small enough exploration rate to guarantee that his long-run average reward converges to the $\varepsilon-$ range of the maximum reward he can get.

6. EXPERIMENTS

In this section, we test our algorithms on the CEG shown in Figure 1.

6.1 MLA

In the first experiment, we test the MLA algorithm and set $\alpha_1 = 0.005$, $\alpha_2 = 0.003$, and $\alpha_3 = 0.01$, where α_i is the learning rate of player i. Figure 3 shows the evolution of probability vectors in four nodes of the game. We only show the probability of taking action l in every node. Since there are only two actions available in every node, the probability of taking action r in every node equals one minus the probability of taking action l. Node s_2 is not included in this figure because only one action is available in s_2 and thus $p(s_2, l) = 1$. As we can see, after about 1000 learning episodes, all players' strategies converge very close to the SPE profile. In the SPE profile, $p(s_4, l)$, the probability of player 1 taking action l in node s_4, equals one. Figure 3 shows that $p(s_4, l)$ actually converges to 0.957. The reason is after $p(s_3, l)$ converges very close to zero, player 1, who owns node s_4, has little chance to visit this node any more.

[2]We call this the local time in node s and do not use episode index t here, because any specific action a is not activated in every episode.

If we decrease the maximum learning rate, the error between the SPE profile and actual strategy profile can decrease as well. In Figure 4, we change the learning rate of player 3 from $\alpha_3 = 0.01$ to 0.002 and keep everything the same as in Figure 3. Compared with the previous experiment, two observations about Figure 4 are clear: First, because of the small learning rate of player 3, the convergence speed of $p(s_3, l)$ is slower. (For the convenience of direct comparison, we put Figure 4 in the same time scale as Figure 3. Even though we cannot see the convergence of $p(s_3, l)$ in Figure 4, it actually converges very close to zero.) Second, $p(s_4, l)$ in Figure 4 converges closer to one compared with the convergence in Figure 3. In Figure 4, $p(s_4, l)$ converges to 0.998, in contrast with 0.957 in Figure 3.

6.2 MQ-Learning

The learning rates of the three players are set to be $\gamma_1 = 0.01$, $\gamma_2 = 0.02$, and $\gamma_3 = 0.01$ and the exploration thresholds are set to be $\sigma_1 = 0.1$, $\sigma_2 = 0.05$, and $\sigma_3 = 0.2$. All the initial q-values are zero. Figure 5 shows the experimental results.

As we can see, the q-value of every node-action pair eventually converges to a neighborhood of the SPE reward of the successive node of this pair. Note that $q(s_4, r)$ converges extremely slowly; the reason is that this specific node-action pair has little chance to be activated. We can accelerate the convergence speed by two ways: First, choose larger exploration thresholds to ensure every node-action pair is sufficiently activated in the learning process. Second, choose larger learning rates to speed up the learning process. However, both acceleration methods have undesired side effects: large exploration thresholds enlarge the $\varepsilon-$range that constraints the expected q-values; large learning rates amplify the variance of q-values.

7. RELATED WORK

Another model of multi-agent learning in games is the stochastic game model. Littman [9] studied how agent can learn to play minimax strategies in zero-sum stochastic games. Hu and Wellman [5] and Littman [10] further extended this model by allowing agents to play general-sum stochastic games. These extensions used Nash equilibrium, which extends the minimax concept for zero-sum games to general-sum games, as the solution concept. Bowling and Veloso [2] studied how agents can learn to play general-sum matrix games. Their algorithm, derived from Singh *et al.* [14], which in turn has its root in evolutionary game theory, can only provably guarantee to converge for very simple matrix games with two players and two actions.

Game theorists are also interested in multi-agent learning in various games. Their efforts on this topic, in response to the often-quoted criticism that the various equilibrium concepts in game theory assume *unbounded* rationality of the players, mainly focus on developing an "alternative explanation that equilibrium arises as the long-run outcome of a process in which less than fully rational players grope for optimality over time." (Fudenberg and Levine [3], page 1.) Our technique can be viewed as an example of this process. Another example is the popular technique of fictitious play. An agent fictitiously plays a game by adopting some statistical method to form beliefs about the strategies of other

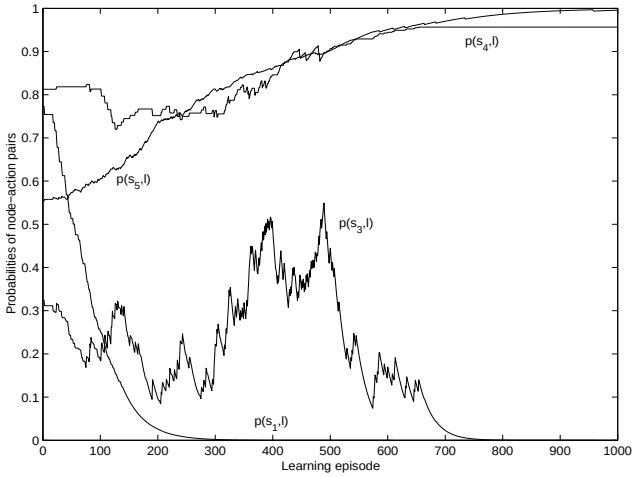

Figure 3: The evolution of three players' strategies when using the automation learning algorithm. The learning rates are set to be $\alpha_1 = 0.005$, $\alpha_2 = 0.003$, and $\alpha_3 = 0.01$.

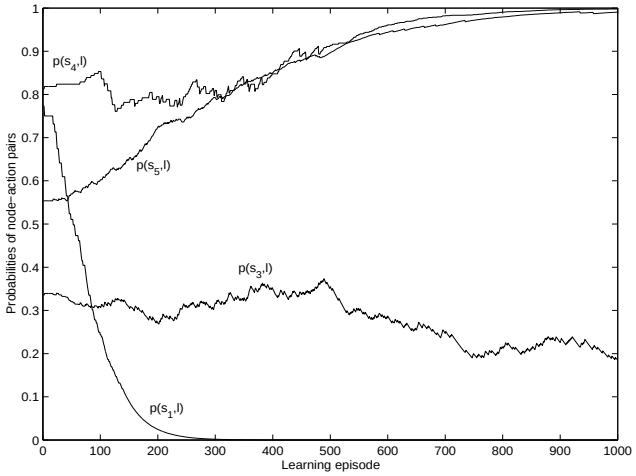

Figure 4: The evolution of three players' strategies when using the automation learning algorithm. Compared with Figure 3, only player 3's learning rate is changed to $\alpha_3 = 0.002$, all other parameters are the same.

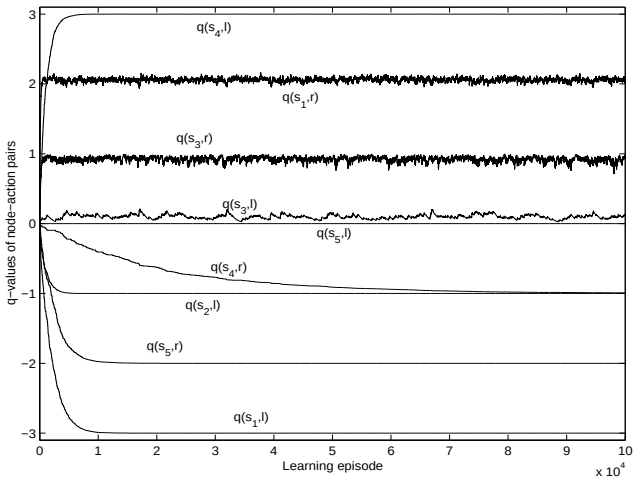

Figure 5: The evolution of q-values when using the MQ-learning algorithm. The exploration thresholds of three players are set to be $\sigma_1 = 0.1$, $\sigma_2 = 0.05$, and $\sigma_3 = 0.2$.

agents and then adjusts his own strategy according to these beliefs. There are potentially many different ways to build beliefs; different ways almost always bring about different learning processes. Therefore, whether, and how, the learning process of a given fictitious-play model converges often heavily depends on how agents build beliefs. Nodleke and Samuelson [12] studied a fictitious play model for CEGs. The learning process in their model does not converge to SPE in general. Actually, the stable points of the learning process in their model consist of not only SPE but also other outcomes. For a complete treatment of fictitious play, refer to Fudenberg and Levine [3] and the references therein.

Recently, reinforcement learning techniques have also drawn attention from game theorists. After finishing this paper, we noticed that Jehiel and Samet [6] proposed a similar algorithm to our MQ-learning. Their algorithm differs from MQ-learning in that it does not use equation (13) to update the q-values and estimates the expected q-values by directly averaging the reward obtained in every episode.

8. CONCLUSIONS

We show that a group of autonomous and self-interested agents can learn to play the SPE strategy in a CEG by repeatedly reinforcing their previous success/failure experience, without building beliefs about each other. This result relies on two factors: (1) the special tree structure of a CEG, and (2) the fact that every move is observable in a CEG.

In an extensive game with incomplete information, a player cannot always observe the previous actions taken by other players. Therefore, a player may not know exactly which tree node he is located at during the learning process. Instead, he only has information about which nodes he might be located at and this information is represented by a probability distribution over all possible nodes. Again, if we assume only one player is learning, the problem of learning in extensive games with incomplete information degenerates to the problem of learning in Partially Observable Markov Decision Processes (POMDPs), with the hidden states corresponding to nodes in the game. In future work, we are going to investigate whether the single-agent learning algorithms [7] for POMDPs can also be extended for learning in extensive games with incomplete information.

Acknowledgment

The authors are grateful to Teddy Seidenfeld, Geoffrey Gordon and three anonymous reviewers for their helpful comments that greatly improved the exposition and content of this paper.

9. REFERENCES

[1] D. P. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, Belmont, MA, 1996.

[2] M. H. Bowling and M. M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

[3] D. Fudenberg and D. K. Levine. *The Theory of Learning in Games.* MIT Press, Cambridge, MA, 1998.

[4] D. Fudenberg and J. Tirole. *Game Theory.* MIT Press, Cambridge, MA, 1991.

[5] J. Hu and M. P. Wellman. Multiagent reinforcement learning: theoretical framework and an algorithm. In *Proc. 15th International Conf. on Machine Learning*, pages 242–250, 1998.

[6] P. Jehiel and D. Samet. Learning to play games in extensive form by evaluation. Working Paper. HTTP://www.tau.ac.il/~samet.

[7] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[8] L. P. Kaelbling, M. L. Littman, and A. P. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[9] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 157–163, 1994.

[10] M. L. Littman. Friend-or-foe q-learning in general-sum games. In *Proceedings of the 18th International Conference on Machine Learning*, pages 322–328, 2001.

[11] S. Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22(1-3):159–195, 1996.

[12] G. Noldeke and L. Samuelson. An evolutionary analysis of backward and forward induction. *Games and Economic Behavior*, 5(3):425–454, 1993.

[13] J. Richard M. Wheeler and K. S. Narendra. Decentralized learning in finite markov chains. *IEEE Transactions on Automatic Control*, AC-31(6):519–526, 1986.

[14] S. Singh, M. Kearns, and Y. Mansour. Nash convergence of gradient dynamics in general-sum games. In *Proc. of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 541–548, 2000.

[15] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, Massachusetts, 1998.