Proc. Indian Acad. Sci. (Math. Sci.), Vol. 104, No. 1, February 1994, pp. 279–294.


# Absolutely expedient algorithms for learning Nash equilibria

V V PHANSALKAR, P S SASTRY and M A L THATHACHAR
Department of Electrical Engineering, Indian Institute of Science, Bangalore 560012, India

Dedicated to the memory of Professor K G Ramanathan

**Abstract.** This paper considers a multi-person discrete game with random payoffs. The distribution of the random payoff is unknown to the players and further none of the players know the strategies or the actual moves of other players. A class of absolutely expedient learning algorithms for the game based on a decentralised team of Learning Automata is presented. These algorithms correspond, in some sense, to rational behaviour on the part of the players. All stable stationary points of the algorithm are shown to be Nash equilibria for the game. It is also shown that under some additional constraints on the game, the team will always converge to a Nash equilibrium.

**Keywords.** Nash equilibria; Decentralised learning algorithm.

## 1. Introduction

This paper is concerned with a learning problem in a general multiperson stochastic game with incomplete information. We study a class of decentralised algorithms for learning Nash equilibria. For this purpose, we employ team concepts associated with Learning Automata models [13].

The game we consider is a discrete stochastic game played by $N$ players. Each of the players has finitely many actions one of which he plays at each instant. After each play, the payoffs to individual players are random variables. The objective for each player is to maximise his expected payoff. Further, the game is one of incomplete information [6]. Thus, nothing is known regarding the distributions of the random payoffs. For learning optimal strategies, the game is palyed repeatedly. We are interested in (asymptotically) learning equilibrium strategies, in the sense of Nash, with respect to the expected value of the payoff. Our interest will be in decentralised learning algorithms. Hence, after each play, each of the players updates his strategy based solely on his current action or move and his payoff. None of the players has any information regarding the other players. As a matter of fact, none of the players need to even know the existence of other players. Thus the game we tackle is also of imperfect information [6].

Such games are useful in tackling problems in many areas such as decentralised control, optimisation, pattern recognition and computer vision. Some of the applications of the game model considered in this paper are discussed in [14]. In many such problems Nash equilibria, in fact, represent the desired solutions. (For a good discussion on the rationality of Nash equilibria see [4, Ch. 2]).

We use a team of learning automata [13] for evolving to the optimal strategies. Games of learning automata have been used as models for adaptive decision making

in many applications [17, 15, 18]. In Learning Automata theory, algorithms for learning optimal strategies have been developed for many special types of finite stochastic games. Some of the models considered are: Two-person zero-sum games [9], $N$-person games with common payoff [17, 16, 20] and non-cooperative games such as Prisoner's Dilemma and Stackelberg games [19]. In [14], it is shown that a team of Learning Automata involved in a general $N$-person stochastic game will converge to a Nash Equilibrium if each of the team members makes use of a linear algorithm called the $L_{R-I}$ algorithm [13]. This requires that every member of the team has to use the same algorithm (though may be with different learning parameters). While this may be useful for applications such as optimization, for general $N$-person games (that include non-cooperative games) this is a restrictive condition. Here, we expand the earlier result [14] to a case where different players use different algorithms (not necessarily linear) though we require that every player satisfy the the 'absolute expediency' property [13]. Informally speaking, the learning algorithm used by a player is absolutely expedient if the algorithm ensures that the expected value of his payoff will increase montonically with time (when all the other players play to a fixed strategy). We feel that this is a reasonable restriction because, assuming rational behaviour, each player should try to increase his payoff.

We begin by formulating the learning problem in §2. Section 3 gives a brief introduction to the necessary ideas from Learning Automata theory. Section 4 presents the learning algorithm and its analysis. Section 5 concludes the paper with a discussion of the results presented in the paper.

## 2. Problem formulation

In this section we introduce our notation and derive a few results regarding Nash equilibria which will be used later on in the analysis of our algorithm. Most of the formulation in this section can be found in any standard book on Game Theory (e.g., [5, 2, 7]).

Consider a $N$-person stochastic game. Each player $i$ has a finite set of actions or pure strategies, $S_i$, $1 \leqslant i \leqslant N$. Let cardinality of $S_i$ be $m_i$, $1 \leqslant i \leqslant N$. (It may be noted that the sets $S_i$, $1 \leqslant i \leqslant N$, need not have any common elements and we assume no structure on these sets). Each play of the game consists of each of the players choosing an action. The result of each play is a random payoff to each player. Let $r_i$ denote the random payoff to player $i$, $1 \leqslant i \leqslant N$. It is assumed that $r_i \in [0, 1]$. Define functions $d^n: \Pi_{j=1}^{N} S_j \rightarrow [0, 1]$, $1 \leqslant i \leqslant N$, by

$$d^n(a_1, \ldots, a_N) = E[r_i | \text{ player } j \text{ chose action } a_j,\ a_j \in S_j,\ 1 \leqslant j \leqslant N] \qquad (1)$$

The function $d^i$ is called the payoff function or utility function of player $i$, $1 \leqslant i \leqslant N$. The objective of each player is to maximise his payoff. A strategy for player $i$ is defined to be a probability vector $\mathbf{q}_i = [q_{i1}, \ldots, q_{im}]^t$, where player $i$ chooses action $j$ with probability $q_{ij}$. The strategy of a player can be time varying as it would be, for example, during learning. Each of the pure strategies or actions of the $i$th player can be considered as a strategy. Let $\mathbf{e}_i$ be a unit probability vector (of appropriate dimension) with $i$th component unity and all others zero. Then $\mathbf{e}_i$ is the strategy corresponding to the action $i$. (It may be noted that any unit probability vector

represents a pure strategy). A strategy that does not necessarily correspond to a pure strategy is called a mixed strategy.

Given the actions chosen by the players, (1) specifies the expected payoff. We can easily extend the functions $d^i$, to the set of all strategies. If $g^i$ is this extension, then it is defined as

$$g^i(\mathbf{q}_1,\ldots,\mathbf{q}_N) = E[r_i|j\text{th player employs strategy } \mathbf{q}_j,\ 1\leqslant j\leqslant N]$$
$$= \sum_{j_1,\ldots,j_N} d^i(j_1,\ldots,j_N)\prod_{s=1}^{N} q_{sj_s} \tag{2}$$

DEFINITION 2.1

*The N-tuple of strategies* $(\mathbf{q}_1^0,\ldots,\mathbf{q}_N^0)$ *is said to be a Nash equilibrium, if for each i,* $1\leqslant i\leqslant N$, *we have*

$$g^i(\mathbf{q}_1^0,\ldots,\mathbf{q}_{i-1}^0,\mathbf{q}_i^0,\mathbf{q}_{i+1}^0,\ldots,\mathbf{q}_N^0)\geqslant g^i(\mathbf{q}_1^0,\ldots,\mathbf{q}_{i-1}^0,\mathbf{q},\mathbf{q}_{i+1}^0,\ldots,\mathbf{q}_N^0)$$
$$\forall \textit{ probability vectors } \mathbf{q}\in[0,1]^{m_i} \tag{3}$$

In general, each $\mathbf{q}_i^0$ above will be a mixed strategy and then we refer to $(\mathbf{q}_1^0,\ldots,\mathbf{q}_N^0)$, satisfying (3), as a Nash equilibrium in mixed strategies. Every $N$-person game will have at least one Nash equilibrium in mixed strategies [4, 5].

We say we have a Nash equilibrium in pure strategies if $(\mathbf{q}_1^0,\ldots,\mathbf{q}_N^0)$ is a Nash equilibrium with each $\mathbf{q}_i^0$ a unit probability vector. In view of (2), for verifying a Nash equilibrium in pure strategies, we can simplify the condition (3) as given in the definition below.

DEFINITION 2.2

*The N-tuple of actions* $(a_1^0,\ldots,a_N^0)$ *(or equivalently the set of strategies* $(\mathbf{e}_{a_1^0},\ldots,\mathbf{e}_{a_N^0})$*) is called a* **Nash equilibrium in pure strategies** *if for each i,* $1\leqslant i\leqslant N$,

$$d^i(a_1^0,\ldots,a_{i-1}^0,a_i^0,a_{i+1}^0,\ldots,a_N^0)\geqslant d^i(a_1^0,\ldots,a_{i-1}^0,a_i,a_{i+1}^0,\ldots,a_N^0),\quad \forall a_i\in S_i. \tag{4}$$

*Here, for all j,* $a_j^0\in S_j$, *the set of pure strategies of player j.*

DEFINITION 2.3

*A Nash equilibrium is said to be* **strict** *if the inequalities in* (3) *(equivalently,* (4)*, for pure strategies) are strict.*

Since each of the sets $S_i$ is finite, each of the functions, $d^i:\Pi_{j=1}^N S_j\to[0,1]$, can be represented by a hyper matrix of dimension $m_1\times\cdots\times m_N$. These $N$ hyper matrices together constitute what can be called the reward structure of the game. Since the game is one of incomplete information these payoff matrices are unknown. Now the learning problem for the game can be stated as follows.

Let $G$ be a $N$-person stochastic game with incomplete information. At any instant $k$, let the strategy employed by the $i$th player be $\mathbf{q}_i(k)$. Let $a_i(k)$ and $r_i(k)$ be the actual actions selected by $i$ and the pay off received by $i$ respectively at $k$, $k=0,1,2,\ldots$. Find a decentralised learning algorithm for the players (that is, design functions $T_i$, where $\mathbf{q}_i(k+1)=T_i(\mathbf{q}_i(k),a_i(k),r_i(k))$) such that $\mathbf{q}_i(k)\to\mathbf{q}_i^0$ as $k\to\infty$ where $(\mathbf{q}_1^0,\ldots,\mathbf{q}_N^0)$ is a Nash equilibrium of the game.

In § 4, we present a team of Learning Automata model for solving this problem after briefly introducing the concept of a Learning Automaton in § 3. In the remaining part of this section, we state a simple result regarding Nash equilibria, which is needed for the analysis later on. Define $\mathbf{K} \subset [0,1]^{m_1+\cdots+m_N}$ by

$$\mathbf{K} = \{Q \in [0,1]^{m_1+\cdots+m_N} : Q = (\mathbf{q}_1, \ldots, \mathbf{q}_N), \text{ and } \forall i, 1 \leqslant i \leqslant N, \mathbf{q}_i \text{ is a } m_i\text{-dimensional probability vector}\} \tag{5}$$

It is easy to see that $\mathbf{K}$ is the set of all $N$-tuples of mixed strategies or the set of possible strategies for the game. Let $\mathbf{K}^* \subset \mathbf{K}$ denote the set of possible pure strategies for the game. $\mathbf{K}^*$ is defined by

$$\mathbf{K}^* = \{Q \in [0,1]^{m_1+\cdots+m_N} : Q = (\mathbf{q}_1, \ldots, \mathbf{q}_N), \text{ and } \forall i, 1 \leqslant i \leqslant N, \mathbf{q}_i \text{ is a } m_i - \text{dimensional probability vector with one component unity}\} \tag{6}$$

It is easy to see that $\mathbf{K}^*$ can be put in one to one correspondence with the set $\Pi_{j=1}^{N} S_j$. Hence we can think of the function $d^i$, given by (1) as defined on $\mathbf{K}^*$. Similarly, functions $g^i$, given by (2), are defined over $\mathbf{K}$. Define functions $h_{is}$, $1 \leqslant s \leqslant m_i$, $1 \leqslant i \leqslant N$, on $\mathbf{K}$ by

$$\begin{aligned} h_{is}(Q) &= E[r_i | \text{ player } j \text{ employed strategy } \mathbf{q}_j,\ 1 \leqslant j \leqslant N,\ j \neq i, \\ &\qquad \text{and player } i \text{ chose action } s] \\ &= \sum_{j_1,\ldots,j_{i-1},j_{i+1},\ldots,j_N} d(j_1,\ldots,j_{i-1},s,j_{i+1},\ldots,j_N) \prod_{t \neq s} q_{tj_t} \end{aligned} \tag{7}$$

where $Q = (\mathbf{q}_1, \ldots, \mathbf{q}_N)$. From (2) and (7), we have

$$\sum_{s=1}^{m_i} h_{is}(Q) q_{is} = g^i(Q) \tag{8}$$

*Lemma* 2.1. *Any* $Q^0 = (\mathbf{q}_1^0, \ldots, \mathbf{q}_N^0) \in \mathbf{K}$ *is a Nash equilibrium if and only if*

$$h_{is}(Q^0) \leqslant g^i(Q^0), \quad \forall s, i.$$

COROLLARY 2.1

*Let* $Q^0 = (\mathbf{q}_1^0, \ldots, \mathbf{q}_N^0)$ *be a Nash equilibrium. Then for each* $i$,

$$h_{is}(Q^0) = g^i(Q^0) \forall s \text{ such that } q_{is}^0 > 0.$$

Both the above results follow directly from the definition of Nash equilibria and are standard results in Game theory (see, for example, [7, Thm. 3.1], and [2, Ch. 3]).

## 3. Learning Automata

In this section, a brief introduction to Learning Automata [13] models is given.

The basic Learning Automata system consists of a Learning Automation interacting

with an environment. At each instant, the Learning Automaton chooses from a finite set of possible actions and outputs it to the environment. The environment then responds with a signal which evaluates the action. This signal is known as the **Scalar Reinforcement Signal** (SRS). This is a real number and a higher value of the SRS indicates better performance of the system. The SRS is assumed to be stochastic in nature. Otherwise each action can be tried once and the action which returns the highest value of the SRS be selected as the optimal action.

The environment is defined by the tuple $\langle A, R, \mathscr{F} \rangle$ where

1. $A$ is the set of actions. It is assumed to be a finite set and

$$A = \{a_1, \ldots, a_m\}.$$

2. $R$ is the set of values the SRS can take. In our case $R$ will be a subset of the closed interval [0, 1]. We denote the value of the SRS at instant $k$ by $r(k)$.
3. $\mathscr{F} = \{F_1, \ldots, F_m\}$ is a set of probability distributions over $R$. $F_i$ is the distribution of the SRS, given that the action taken by the system is $a_i$.

When the $F_i$'s are independent of time, the environment is said to be a stationary environment. We consider such environments in this section. Define

$$d_i = E^i(r)$$

where $E^i$ denotes expectation with respect to $F_i$. Thus, $d_i$ is the expected value of the SRS if action $a_i$ is the output. The optimal action is defined as the action which has the highest expected value of the SRS. It is seen that the optimal action is defined independently of the system used to learn it. Thus, the problem is completely defined by the environment.

The basic model of the learning automaton maintains a probability distribution over the set of actions $A$. The notation used is the same as the used in describing the environment above. The number of actions is $m$ and the probability distribution is a $m$-dimensional real vector $p$ such that

$$p = (p_1, \ldots, p_m)^t$$

$$p_i \geqslant 0 \quad \forall i \quad 1 \leqslant i \leqslant m$$

$$\sum_{i=1}^{m} p_i = 1$$

At instant $k$, if $p(k)$ is the action probability vector, the probability of choosing the $i$th action is $p_i(k)$. Formally, a learning automaton is defined by the tuple $\langle A, Q, R, T \rangle$ where

1. $A$ is the (finite) set of actions available to the automaton.
2. $Q$ is the state of the learning automaton. In standard learning automata theory, $Q$ is the action probability vector $p$. In estimator algorithms [17], $Q$ consists of the action probability vector along with a vector of estimates. In the algorithms considered in this paper $Q = p$. Hence we use $p$ for the state of the learning automaton.
3. $R$ is the set from which the SRS takes its values. It is the same as $R$ in the definition of the environment.

4. $T$ is the learning algorithm which updates $p$.

$$p(k+1) = T(p(k), r(k), a(k))$$

$a(k)$ is the action of the automation at instant $k$.

The optimal action is defined to be that which has the highest value of $d_i$. One method of evaluating the performance of the system is to check whether the probability of choosing the optimal action becomes unity asymptotically. This is difficult to ascertain and various other performance measures have been defined [13]. They are briefly described below.

Any algorithm should at least do better than a method which chooses actions randomly (with equal probability) at each instant. An automaton which uses this technique is called the pure chance automaton. For comparing the performance of automata with the pure chance automaton, the average value of the SRS at a state $p$ is needed. This is denoted by $M(k)$. Thus

$$M(k) = E[r(k)|p(k)]$$

For the pure chance automaton, this quantity is a constant, denoted by $M_0$.

$$M_0 = \frac{1}{m} \sum_{i=1}^{m} d_i$$

An automaton should at least do better than $M_0$. This property is known as expediency.

DEFINITION 3.1

*A learning automaton is said to be expedient if*

$$\liminf_{k \to \infty} E[M(k)] > M_0$$

This property does not say much. What is really needed is that the automaton converge to the optimal action. Let the actions of the automaton be $\{a_1, \ldots, a_m\}$. Let $a_s$ be the unique optimal action. That is, $d_s > d_i$ for all $i \neq s$.

DEFINITION 3.2

*A learning automaton is said to be optimal if*

$$\lim_{k \to \infty} E[M(k)] = d_s$$

Optimality is not an easy property to prove. In fact, no algorithm has this property. A weaker property is that of $\varepsilon$-optimality. This says that even though the optimal value $d_s$ cannot be achieved, it should be possible to approach it within any prespecified accuracy.

DEFINITION 3.3

*A learning automaton is said to be ε-optimal if*

$$\liminf_{k\to\infty} E[M(k)] > d_s - \varepsilon$$

*is achieved for any* $\varepsilon > 0$ *by an appropriate choice of the automaton parameters.*

In general, different values of the automaton parameters would be required for different values of ε.

A property which can be checked without asymptotic analysis and has been shown to imply ε-optimality in stationary environments [13] is absolute expediency.

DEFINITION 3.4

*A learning automaton is said to be absolutely expedient if*

$$E[M(k+1)|p(k)] > M(k)$$

*for all k, all probabilities in the open interval* (0, 1) *and all stationary environments with a unique optimal action.*

Necessary and sufficient conditions for an algorithm to be absolutely expedient were first given in [10] and generalised in [1].

Various algorithms have been developed for learning automata. The class of algorithms considered in this paper is those of absolutely expedient algorithms. These algorithms are described below.

$p(k)$ is the action probability vector at instant $k$. Let $a(k)$ denote the action at instant $k$ and $r(k)$ the SRS. It is necessary that the SRS take values from a subset of the closed interval [0, 1]. The general form of the absolutely expedient algorithm is

If the action chosen at instant $k$ is $a(k) = a_j$ then

$$\begin{aligned} p_s(k+1) &= p_s(k) - br(k)\alpha_{js}(p(k)) + b(1-r(k))\beta_{js}(p(k)) \quad s \neq j \\ p_j(k+1) &= p_j(k) + br(k)\sum_{s\neq j}\alpha_{js}(p(k)) - b(1-r(k))\sum_{s\neq j}\beta_{js}(p(k)) \end{aligned} \tag{9}$$

where $0 < b < 1$ is the learning parameter. The following conditions are imposed on the $\alpha$ and $\beta$ functions so that $p(k+1)$ remains a probability vector.

$$\begin{aligned} &\alpha_{js}(p) \leqslant p_s \quad \forall j, s \\ &\sum_{s\neq j}\beta_{js}(p) \leqslant p_j \quad \forall j \end{aligned} \tag{10}$$

Necessary and sufficient conditions for this algorithm to be absolutely expedient were given by Aso-Kimura [1]. These are

$$\begin{aligned} \sum_{s\neq j} p_s\alpha_{sj} &= \sum_{s\neq j} p_j\alpha_{js} \quad \forall j \\ \sum_{s\neq j} p_s\beta_{sj} &= \sum_{s\neq j} p_j\beta_{js} \quad \forall j \end{aligned} \tag{11}$$

If we set $\alpha_{js} = p_s$ and $\beta_{js} = 0$ we get the $L_{R-I}$ algorithm mentioned earlier. This is the algorithm considered in [14] for the Game problem. The algorithm is easily seen to satisfy conditions (11).

Absolutely expedient algorithms have been shown to be $\varepsilon$-optimal with respect to the learning parameter $b$ [11]. The first conditions for absolute expediency were given in [10]. In these class of algorithms, $\alpha_{js}(p) = \alpha_s(p)$ and $\beta_{js}(p) = \beta_s(p)$. Necessary and sufficient conditions for these class of algorithms to be absolutely expedient are

$$\begin{aligned} &\alpha_j/p_j = \lambda(p) \quad \forall j \\ &\beta_j/p_j = \mu(p) \quad \forall j \end{aligned} \tag{12}$$

A set of conditions which are easily seen to be more general than the (12) conditions but more restrictive than the (11) conditions are

$$\begin{aligned} &p_s\alpha_{sj} = p_j\alpha_{js} \quad \forall j, s \quad s \neq j \\ &p_s\beta_{sj} = p_j\beta_{js} \quad \forall j, s \quad s \neq j \end{aligned} \tag{13}$$

$\alpha$ and $\beta$ which satisfy (13) but not (12) are

$$\alpha_{js} = \beta_{js} = p_j p_s^2 \quad \forall j, s, \quad s \neq j \tag{14}$$

The following simple lemma will be needed in the the next section.

*Lemma* 3.1 *If* $p_j = 0$ *or* $p_s = 0$, *and the Aso-Kimura conditions* (11) *are satisfied, then*

$$p_s(\alpha_{sj} + \beta_{sj}) = 0 \tag{15}$$

*Proof.* Trivially, condition (15) is satisfied if $p_s = 0$. Let $p_s \neq 0$ and $p_j = 0$. As $\alpha_{sj} \leqslant p_j$, $\alpha_{sj} = 0$. By the Aso-Kimura conditions (11),

$$\begin{aligned} \sum_{a \neq j} p_a\beta_{aj} &= \sum_{a \neq j} p_j\beta_{ja} \\ &= 0 \text{ as } p_j = 0. \end{aligned}$$

Thus $\Sigma_{a \neq j} p_a\beta_{aj} = 0$ and as each term is non-negative, $p_a\beta_{aj} = 0$ for all $a$. In particular, $p_s\beta_{sj} = 0$. ■

It is also seen from the above proof that

$$p_j = 0 \Rightarrow (\alpha_{sj} + \beta_{sj}) = 0. \tag{16}$$

An additional condition which will be used in certain proofs in the next section is the following

$$p_j \neq 0 \Rightarrow (\alpha_{sj} + \beta_{sj}) > 0. \tag{17}$$

This is satisfied, for example, by the $L_{R-I}$ algorithm.

## 4. Algorithm and analysis

We consider an $N$-person game where the $i$th player has $m_i$ pure strategies. We will represent each player by a learning automaton and the actions of the automaton are the pure strategies of the player. Let $\mathbf{p}_i(k) = [p_{i1}(k) \cdots p_{im_i}(k)]^t$ denote the action probability distribution of the $i$th player. $p_{ij}(k)$ denotes the probability with which $i$th automaton player chooses the $j$th pure strategy at instant $k$. Thus $\mathbf{p}_i(k)$ is the strategy employed by the $i$th player at instant $k$. Each play of the game consists of each of the automata players choosing an action independently and at random according to their current action probabilities. The payoff to the $i$th player will be the reaction to the $i$th automaton which will be denoted by $r_i(k)$. The learning algorithm used by each of the player is as given below.

1. At each instant $k$, player $i$ selects an action from his action set $S_i$ according to his current action probability vector $p_i(k)$. Thus, if $a_i(k)$ is the action at instant $k$ and $S_i = \{a_{i1}, \ldots, a_{im_i}\}$, then

$$\text{Prob}(a_i(k) = a_{ij}) = p_{ij}(k).$$

2. Based on the actions taken by all the players, player $i$ receives a payoff $r_i(k)$ given by (1).
3. Let the action of the $i$th player at instant $k$ be $a_{ij}$. Every player updates his action probabilities according to the following rule.

$$\begin{aligned} p_{is}(k+1) &= p_{is}(k) - br_i(k)\alpha_{ijs}(\mathbf{p}_i(k)) + b(1 - r_i(k))\beta_{ijs}(\mathbf{p}_i(k)) \quad s \neq j \\ p_{ij}(k+1) &= p_{ij}(k) + br_i(k)\sum_{s \neq j} \alpha_{ijs}(\mathbf{p}_i(k)) - b(1 - r_i(k))\sum_{s \neq j} \beta_{ijs}(\mathbf{p}_i(k)) \end{aligned} \tag{18}$$

where $0 < b < 1$ is the learning parameter. For simplicity of notation, it is assumed here that all players use the same value of $b$. All the results would hold even if different values of the learning parameter were used by the players. $\alpha_{ijs}$, $\beta_{ijs}$ are the functions used by player $i$ to update his strategy and are analogous to $\alpha_{js}$ and $\beta_{js}$ functions for the single automaton described in the previous section. It may be noted that different players may use different functions to update their strategies provided the functions satisfy the conditions described below.

The $\alpha$ and $\beta$ functions satisfy the Aso-Kimura conditions for absolute expediency. They are

$$\begin{aligned} \sum_{s \neq j} p_{is}\alpha_{isj} &= \sum_{s \neq j} p_{ij}\alpha_{ijs} \\ \sum_{s \neq j} p_{is}\beta_{isj} &= \sum_{s \neq j} p_{ij}\beta_{ijs} \end{aligned} \tag{19}$$

$\alpha_{ijs}$ and $\beta_{ijs}$ are non-negative and satisfy conditions (10) to keep $\mathbf{p}_i(k+1)$ an action probability vector. They can be written as

$$\begin{aligned} \alpha_{ijs}(\mathbf{p}_i) &\leqslant p_{is} \quad \forall i, j, s. \quad s \neq j \\ \sum_{s \neq j} \beta_{ijs}(\mathbf{p}_i) &\leqslant p_{ij} \quad \forall i, j \end{aligned} \tag{20}$$

In addition to these conditions an additional condition is imposed to ensure that the updating is not stopped prematurely. This is

$$\alpha_{ijs} + \beta_{ijs} \neq 0 \text{ if } p_{ij} \neq 0 \text{ and } p_{is} \neq 0. \tag{21}$$

4.1 *Analysis of the algorithm*

The analysis of the algorithm is carried out in two stages. First, weak convergence techniques are used to show that the algorithm can be approximated by an appropriate ODE (Ordinary Differential Equation) as $b \to 0$. Then, solutions of the ODE are analysed to obtain information about the behaviour of the algorithm.

The learning algorithm given by (18) can be represented as

$$P(k+1) = P(k) + bG(P(k), a(k), r(k)) \tag{22}$$

where $a(k) = (a_1(k) \cdots a_N(k))$ denotes the actions selected by the automata team at $k$ and $r(k) = (r_1(k) \cdots r_N(k))$ are the resulting payoffs.

Let $P(k) = (\mathbf{p}_1(k), \ldots, \mathbf{p}_N(k))$ denote the vector current mixed strategies of all the players which is also the state of the learning algorithm at instant $k$. Our interest is in the asymptotic behaviour of $P(k)$. Since each $\mathbf{p}_i$ is a probability vector, we have $P(k) \in \mathbf{K}$ where $\mathbf{K}$ is as defined by (5). The following piecewise-constant interpolation of $P(k)$ is required to use the weak convergence techniques

$$P^b(t) = P(k), \quad t \in [kb, (k+1)b) \tag{23}$$

$P^b(\cdot) \in D^{m_1 + \cdots + m_N}$, the space of all functions from $\mathbb{R}$ into $[0,1]^{m_1 + \cdots + m_N}$, which are continuous on the right and have limits on the left. (It may be noted that $P^b(t) \in \mathbf{K}, \forall t$). Now consider the sequence $\{P^b(\cdot), b > 0\}$. We are interested in the limit of this sequence as $b \to 0$.

Define a function $\xi : \mathbf{K} \to [0,1]^{m_1 + \cdots + m_N}$ by

$$\xi(P) = E[G(P(k), a(k), r(k)) | P(k) = P] \tag{24}$$

**Theorem 4.1.** *The sequence of interpolated processes $\{P^b(\cdot)\}$ converges weakly, as $b \to 0$, to $X(\cdot)$ which is the solution of the ODE,*

$$\frac{dX}{dt} = \xi(X), \quad X(0) = P_0 \tag{25}$$

*Proof.* The following conditions are satisfied by the learning algorithm given by (22)

1. $\{P(k), (a(k-1), r(k-1)), k \geq 0\}$ is a Markov process. $(a(k), r(k))$ take values in a compact metric space.
2. The function $G(.,.,.)$ is bounded and continuous and independent of $b$.
3. If $P(k) \equiv P$, then $\{(a(k), r(k)), k \geq 0\}$ is an i.i.d. sequence. Let $M^P$ denote the (invariant) distribution of this process.
4. The ODE (25) has a unique solution for each initial condition.

Hence by [8, Thm. 3.2], the sequence $\{P^b(\cdot)\}$ converges weakly as $b \to 0$ to the solution of the ODE,

$$\frac{dX}{dt} = \bar{G}(X), X(0) = P_0$$

where $\bar{G}(P) = E^P G(P(k), a(k), r(k))$ and $E^P$ denotes the expectation with respect to the invariant measure $M^P$.

Since for $P(k) \equiv P$, $(a(k), r(k))$ is i.i.d. whose distribution depends only on $P$ and the payoff matrices,

$$\bar{G}(P) = E[G(P(k), a(k), r(k)) | P(k) = P] = \xi(P), \quad \text{by (24)}$$

Hence the theorem. ■

*Remark* 4.1. The convergence of functionals implied by weak convergence, along with the knowledge of the nature of the solutions of the ODE (25), enables one to understand the long-term behaviour of $P(k)$. It can be shown that $P(k)$ follows the trajectory $X(t)$ of the ODE within a prespecified error and for as long as prespecified with probability increasingly close to 1 as $b$ decreases. (See [3, Chapter 2, Theorem 1] and the discussion therein).

The following lemma gives an explicit characterisation of $\xi$.

*Lemma* 4.1

$$\xi_{ij}(P) = \sum_{s \neq j} p_{is}(\alpha_{isj} + \beta_{isj})(h_{ij} - h_{is}) \tag{26}$$

*where* $h_{is}$ *are as defined by* (7).

*Proof.* Let $G_{ij}$ denote the $(i,j)$th component of $G$.

$$\begin{aligned}
\xi_{ij}(P) &= E[G_{ij}(P(k), a(k), r(k) | P(k) = P] \\
&= \sum_s E[G_{ij}(P(k), a(k), r(k) | P(k) = P, a_i(k) = a_{is}] p_{is} \\
&= \sum_{s \neq j} E[r_i(k)\alpha_{ijs}(\mathbf{p}_i(k)) - (1 - r_i(k))\beta_{ijs}(\mathbf{p}_i(k)) | P(k) = P, a_i(k) = a_{ij}] p_{ij} \\
&\quad + \sum_{s \neq j} E[-r_i(k)\alpha_{isj}(\mathbf{p}_i(k)) + (1 - r_i(k))\beta_{isj}(\mathbf{p}_i(k)) | P(k) \\
&\quad\quad = P, a_i(k) = a_{is}] p_{is} \\
&= \sum_{s \neq j} \{p_{ij}\alpha_{ijs}(\mathbf{p}_i) E[r_i | P, a_{ij}] - p_{ij}\beta_{ijs}(\mathbf{p}_i) E[1 - r_i | P, a_{ij}] p_{ij}\} \\
&\quad + \sum_{s \neq j} \{-p_{is}\alpha_{isj}(\mathbf{p}_i) E[r_i | P, a_{is}] - p_{is}\beta_{isj}(\mathbf{p}_i) E[1 - r_i | P, a_{is}] p_{is}\} \\
&= \sum_{s \neq j} p_{ij}(\alpha_{ijs} h_{ij} - \beta_{ijs}(1 - h_{ij})) + \sum_{s \neq j} -p_{is}(\alpha_{isj} h_{is} - \beta_{isj}(1 - h_{is})) \\
&= \sum_{s \neq j} p_{is}(\alpha_{isj} + \beta_{isj})(h_{ij} - h_{is}) \text{ by the Aso-Kimura conditions (19)}
\end{aligned}$$

completing the proof. ■

Using (26), the ODE (25) can be written as

$$\frac{dp_{ij}}{dt} = \sum_{s \neq j} p_{is}(\alpha_{isj}(\mathbf{p}_i) + \beta_{isj}(\mathbf{p}_i))[h_{ij}(P) - h_{is}(P)] \quad 1 \leqslant j \leqslant m_i, \quad 1 \leqslant i \leqslant N. \tag{27}$$

The following theorem characterises the solutions of the ODE and hence the long-term behaviour of the algorithm.

**Theorem 4.2** *The following are true of the ODE (and hence of the learning algorithm if the parameter b in* (18) *is sufficiently small).*

1. *All corners of* $\mathbf{K}$ *(i.e. points in* $\mathbf{K}^*$*) are stationary points.*
2. *All Nash equilibria are stationary points.*
3. *If conditions* (13) *and* (17) *are satisfied, all stationary points that are not Nash equilibria are unstable.*
4. *All corners of* $\mathbf{K}$ *that are strict Nash equilibria are locally asymptotically stable.*

*Proof.* 1. Let $P^0 \in \mathbf{K}^*$. Thus $p_{ij}^0 = 0$ or $p_{is}^0 = 0$ for all $j, s$ such that $j \neq s$ and $1 \leqslant j, s \leqslant m_i$.
By Lemma 3.1, $p_{is}^0(\alpha_{isj} + \beta_{isj}) = 0$ for all $s$. Thus,

$$\frac{dp_{ij}}{dt} = \sum_{s \neq j} p_{is}(\alpha_{isj}(\mathbf{p}_i^0) + \beta_{isj}(\mathbf{p}_i^0))[h_{is}(P^0) - h_{is}(P^0)] = 0$$

2. Let $P^0$ be a Nash equilibrium. Define

$$A_i = \{s : p_{is}^0 > 0\}$$

For $j \notin A_i$, $p_{ij}^0 = 0$. Thus, $p_{is}^0(\alpha_{isj} + \beta_{isj}) = 0$ for all $s \neq j$.
Let $j \in A_i$, implying $p_{ij}^0 > 0$. It is trivially seen that $p_{is}^0(\alpha_{isj} + \beta_{isj})(h_{ij} - h_{is}) = 0$ if $p_{is}^0 = 0$. Therefore let $p_{is}^0 > 0$. But then as $P^0$ is a Nash equilibrium, $h_{ij}(P^0) = h_{is}(P^0)$ by Lemma 2.1 Thus

$$p_{is}^0(\alpha_{isj} + \beta_{isj})(h_{ij} - h_{is}) = 0 \quad \forall i, j, s \quad (s \neq j)$$

Thus $P^0$ is a stationary point.
3. Let $P^0$ be a zero of $\xi(\cdot)$ which is not a Nash equilibrium. It is assumed that conditions (13) and (17) are satisfied by the algorithm. By Lemma 2.1, there is an $i$ and an $s$ such that

$$h_{is}(P^0) > g^i(P^0) \tag{28}$$

In general, there will be more than one $s$ such that $h_{is}(P^0) > g^i(P^0)$. Without loss of generality let

$$h_{i1}(P^0) = h_{i2}(P^0) = \cdots = h_{iL}(P^0) > h_{iL+1}(P^0) \geqslant h_{iL+2}(P^0) \geqslant \cdots$$

where $h_{is}(P^0) > g^i(P^0)$, $1 \leqslant s \leqslant L$. Then, for all $\delta$, $\varepsilon$ sufficiently small, there exists a neighbourhood $\mathcal{U}_{\varepsilon\delta}$ around $P^0$ such that for all $P \in \mathcal{U}_{\varepsilon\delta}$, $h_{ij}(P) - h_{is}(P) > \varepsilon$ for all $j \leqslant L$, $s > L$ and $h_{ij}(P) - h_{is}(P) > -\delta\varepsilon$ for all $s, j \leqslant L$. Then

$$\frac{dp_{i1}}{dt} = \sum_{s\neq 1} p_{is}(\alpha_{is1} + \beta_{is1})(h_{i1} - h_{is})$$

$$= p_{i1} \sum_{s\neq 1} (\alpha_{i1s} + \beta_{i1s})(h_{i1} - h_{is}) \text{ as (13) is satisfied}$$

$$= p_{i1} \sum_{s>L} (\alpha_{i1s} + \beta_{i1s})(h_{i1} - h_{is}) + p_{i1} \sum_{2\leqslant s\leqslant L} (\alpha_{i1s} + \beta_{i1s})(h_{i1} - h_{is})$$

$$> \varepsilon p_{i1} \left( \sum_{s>L} (\alpha_{i1s} + \beta_{i1s}) - \delta \sum_{2\leqslant s\leqslant L} (\alpha_{i1s} + \beta_{i1s}) \right) \quad \text{if } P \in \mathscr{U}_{\varepsilon\delta}$$

There is at least one $s > L$ such that $p_{is}^0 > 0$, as $P^0$ is not a Nash equilibrium. Thus, $\alpha_{i1s}(P^0) + \beta_{i1s}(P^0) > 0$, by (17). Thus the linearised version of the last line in the above equation is strictly positive and thus $P^0$ is not stable.

It should be possible to relax condition (17), but the analysis would then involve higher order terms instead of just linear terms. But the same result should go through.

4. Let $P^0 = (\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_N})$ be a corner of $\mathbf{K}$ that is a Nash equilibrium. Without loss of generality, let $i_1 = i_2 = \cdots = i_N = 1$. Use the transformation $P \to \varepsilon$, defined by

$$\varepsilon_{iq} = p_{iq} \text{ if } q \neq 1$$

$$\varepsilon_{i1} = 1 - p_{i1}$$

Define a Lyapunov function $V(\cdot)$ as

$$V(\varepsilon) = \sum_{i=1}^{N} \varepsilon_{i1}$$

It is easy to see that $V \geqslant 0$ and that $V = 0$ iff $\varepsilon = 0$. Also, as $P^0$ is a strict Nash equilibrium, $h_{i1}(P^0) > h_{is}(P^0)$ for all $s$. Thus,

$$\frac{dV}{dt} = \sum_{i=1}^{N} \frac{d\varepsilon_{i1}}{dt}$$

$$= -\sum_{i=1}^{N} \frac{dp_{i1}}{dt}$$

$$= -\sum_{i=1}^{N} \sum_{s\neq 1} p_{is}(\alpha_{is1} + \beta_{is1})(h_{i1} - h_{is})$$

$$= -\sum_{i=1}^{N} \sum_{s\neq 1} \varepsilon_{is}(\alpha_{is1}(\mathbf{p}_i^0) + \beta_{is1}(\mathbf{p}_i^0))(h_{i1}(P^0) - h_{is}(P^0))$$

$$+ \text{higher order terms in } \varepsilon$$

$$< 0 \text{ as } h_{i1}(P^0) > h_{is}(P^0) \quad \forall s \neq 1$$

This proves $P^0$ is asymptotically stable. ■

*Remark* 4.2. Because of the above theorem, we can conclude that our learning algorithm will never converge to a point in $\mathbf{K}$ which is not a Nash equilibrium and

strict Nash equilibria in pure strategies are locally asymptotically stable. This still leaves two questions unanswered. (i) Do Nash equilibria in mixed strategies form stable attractors for the algorithm, and (ii) is it possible that $P(k)$ does not converge to a point in $\mathbf{K}$ which would be the case, for example, if the algorithm exhibits limit cycle behaviour. At present we have no results concerning the first question. Regarding the second question we provide a sufficient condition for $P(k)$ to converge to some point in $\mathbf{K}$. This is proved in Theorem 4.3 below.

**Theorem 4.3.** *Suppose there is a bounded differentiable function*

$$F: \mathbb{R}^{m_1+\cdots+m_N} \to \mathbb{R}$$

*such that for some constants* $c_i > 0$,

$$\frac{\partial F}{\partial p_{iq}}(P) = c_i h_{iq}(P), \quad \forall i, q \text{ and all } P \in \mathbf{K}. \tag{29}$$

*Further, the* $\alpha$ *and* $\beta$ *functions satisfy condition* (13) *and* (17). *Then the learning algorithm, for any initial condition in* $\mathbf{K} - \mathbf{K}^*$, *always converges to a Nash equilibrium.*

*Proof.* Consider the variation of $F$ along the trajectories of the ODE. We have

$$\begin{aligned}
\frac{dF}{dt} &= \sum_{i,q} \frac{\partial F}{\partial p_{iq}} \frac{dp_{iq}}{dt} \\
&= \sum_{i,q} \frac{\partial F}{\partial p_{iq}}(P) \sum_s p_{is}(\alpha_{isq} + \beta_{isq})[h_{iq}(P) - h_{is}(P)], \quad \text{by (26)} \\
&= \sum_i c_i \sum_q \sum_s p_{is}(\alpha_{isq} + \beta_{isq})[(h_{iq}(P))^2 - h_{iq}(P)h_{is}(P)], \quad \text{by (29)} \\
&= \sum_i c_i \sum_q \sum_{s>q} p_{is}(\alpha_{isq} + \beta_{isq})[h_{iq}(P) - h_{is}(P)]^2 \\
&\geqslant 0
\end{aligned} \tag{30}$$

Thus $F$ is nondecreasing along the trajectories of the ODE. Also, due to the nature of the learning algorithm given by (18), all solutions of the ODE (25), for initial conditions in $\mathbf{K}$, will be confined to $\mathbf{K}$ which is a compact subset of $\mathbb{R}^{m_1+\cdots+m_N}$. Hence by [12, Theorem 2.7], asymptotically all the trajectories will be in the set $\mathbf{K}_1 = \{P \in [0,1]^{m_1+\cdots+m_n} : dF/dt(P) = 0\}$.

From (30), (25) and (26) it is easy to see that

$$\begin{aligned}
\frac{dF}{dt}(P) = 0 &\Rightarrow p_{is}(\alpha_{isq} + \beta_{isq})[h_{iq}(P) - h_{is}(P)] = 0 \quad \forall q, s, i \\
&\Rightarrow \xi_{iq}(P) = 0 \quad \forall i, q \\
&\Rightarrow P \text{ is a stationary point of the ODE.}
\end{aligned}$$

Thus all solutions have to converge to some stationary point. Since by Theorem 4.2 all stationary points that are not Nash equilibria are unstable, the theorem follows. ■

Theorem 4.2, and Theorem 4.3 together characterise the long-term behaviour of the learning algorithm. For any general $N$-person game, all strict Nash equilibria in pure strategies are asymptotically stable in the small. Further the algorithm cannot converge to any point in **K** which is not a Nash equilibrium. If the game satisfies the sufficiency condition needed for Theorem 4.3 then the algorithm will converge to a Nash equilibrium. (If the game does not satisfy this condition we cannot be sure that the algorithm will converge rather than, e.g., exhibit a limit cycle behaviour). We have not been able to establish that, in a general game, all mixed strategy equilbria are stable attractors.

## 5. Discussion

In this paper we have considered an $N$-person stochastic game with incomplete information. We presented a method based on Learning Automata for the players to learn equilibrium strategies in a decentralised fashion. In our framework, each player can choose his own learning algorithm. However, the algorithm of each player should satisfy the so called absolute expediency property. In the context of a learning automation, an algorithm is absolutely expedient if the expected reinforcement will increase monotonically with time in all stationary random environments. In the context of the Game, since all the players are updating their strategies, the effective environment as seen by a player will be nonstationary. Here, the restriction of absolute expediency will mean that the algorithm used by a player should ensure the expected payoff to the player increases monotonically when all other players are using fixed strategies. Thus, this is a mild requirement because rational behaviour on the part of the player demands that, at the minimum, he should strive to improve his payoff when everyone else is playing to a fixed strategy. The analysis presented in §4 tells us that if all players are using absolutely expedient algorithms, they can converge to Nash equilibria without needing any information exchange.

In a truly competitive game situation, we may require that the learning algorithm employed by a player should also help to confuse the opponent. The framework presented in this paper does not address this aspect. The analysis presented here also does not address the question of how a player, using an absolutely expedient algorithm, will perform if other players are using arbitrary learning algorithms that are not absolutely expedient. However, in many cases, game models are employed as techniques for solving certain type of optimisation problems. Examples include learning optimal discriminant functions in Pattern Recognition and the Consistent Labelling problem in Computer Vision [17, 18]. (See [14] for a discussion on the application of the Game model considered here in such problems. In such applications the sufficiency condition of Theorem 4.3 also holds). In all such situations, the requirement that the players use absolutely expedient learning algorithms is not restrictive.

Our algorithm can be used for learning the Nash equilibrium even if the game is deterministic and the game matrix is known. We then simply make $r_i$, payoff to the $i$th player, equal to (suitably normalised) $d^i(a_1, \cdots, a_N)$ which is the game matrix entry corresponding to the actions played. Whether this is an efficient algorithm for obtaining Nash equilibria for general deterministic games with known payoff functions needs to be investigated.

Our analysis does not establish the stability or otherwise of Nash equilibria in mixed strategies for the general $N$-person game. In the context of a single Learning Automaton, it is known that in any stationary random environment absolutely expedient algorithms always converge to a unit vector [13] and hence it might be the case that even the decentralised team cannot converge to an interior point. This aspect needs further investigation.

## Acknowledgement

This work was supported by the Office of Naval Research Grant No. 00014-92-J-1324 under an Indo-US project.

## References

[1] Aso H and Kimura M, Absolute expediency of learning automata. *Inf. Sci.* **17** (1976) 91–122
[2] Basar T and Olsder G J, *Dynamic noncooperative game theory*, (New York: Academic Press) (1982)
[3] Albert Beneveniste, Michel Metivier and Pierre Priouret, *Adaptive algorithms and stochastic approximations.* (New York: Springer Verlag) (1987)
[4] Binmore, *Essays on foundations of game theory.* (Basil Blackwell) (1990)
[5] Friedman J W, *Oligopoly and the theory of games.* (New York: North Holland) (1977)
[6] Harsanyi J C, Games with incomplete information played by bayesian player – I, *Manage. Sci.* **14** (1967) 159–182
[7] Wang Jianhua, *The theory of games.* (Oxford: Clarendon Press) (1988)
[8] Kushner H J, *Approximation and weak convergence methods for random processes* (Cambridge MA: MIT Press) (1984)
[9] Lakshmivarahan S and Narendra K S, Learning algorithms for two-person zero-sum stochastic games with incomplete information: a unified approach, *SIAM J. Control Optim.* **20** (1982) 541–552
[10] Lakshmivarahan S and Thathachar M A L, Absolutely expedient learning algorithms for stochastic automata. *IEEE Trans. Syst., Man Cybern.* **3** (1973) 281–286
[11] Meybodi M R and Lakshmivarahan S, $\varepsilon$-optimality of a general class of absorbing barrier learning algorithms, *Inf. Sci.* **28** (1982) 1–20
[12] Narendra K S and Annaswamy A, *Stable adaptive systems*, (Englewood Cliffs: Prentice Hall) (1989)
[13] Narendra K S and Thathachar M A L, *Learning automata: An introduction.* (Englewood Cliffs: Prentice Hall) (1989)
[14] Sastry P S, Phansalkar V V and Thathachar M A L, Decentralized learning of Nash equilibria in multi-person stochastic games with incomplete information. To appear in *IEEE Tran. Syst., Man Cybern.*
[15] Srikanta Kumar P R and Narendra K S, A learning model for routing in telephone networks, *SIAM J. Control Optim.* **20** (1982) 34–57
[16] Thathachar M A L and Ramakrishnan K R, A cooperative game of a pair of automata. *Automatica* **20** (1984) 797–801
[17] Thathachar M A L and Sastry P S, Learning optimal discriminant functions through a cooperative game of automata, *IEEE Trans. Syst., Man Cybern.* **17(1)** (1957) 73–85
[18] Thathachar M A L and Sastry P S, Relaxation labelling with learning automata, *IEEE Trans. Pattern Anal. Mach. Intelligence*, **8** (1986) 256–268
[19] Vishwanatha Rao T, *Learning solutions to stochastic non-cooperative games*, ME Thesis, (Dept. of Electrical Engineering, Indian Institute of Science), Bangalore, India (1984)
[20] Wheeler Jr R M, and Narendra K S, Decentralized learning in finite markov chains, *IEEE Trans. Autom. Control.* **31(6)** (1986) 519–526