DISCRETIZED PURSUIT LINEAR REWARD-INACTION LEARNING AUTOMATA

B.J. Oommen and Joseph K. Lanctôt

SCR-TR-157, April 1989

* Partially supported by the Natural Sciences and Engineering Research Council of Canada. A preliminary version of this paper will be presented as an invited paper at the 1989 IEEE International Conference on Systems, Man and Cybernetics, November 1989 in Boston, Mass.

School of Computer Science, Carleton University
Ottawa, Canada, KIS 5B6

DISCRETIZED PURSUIT LINEAR REWARD-INACTION LEARNING AUTOMATA*

B. J. Oommen and Joseph K. Lanctôt
School of Computer Science
Carleton University
Ottawa : ONT : K1S 5B6
CANADA

ABSTRACT

We consider the problem of a stochastic learning automaton interacting with an unknown random environment. The fundamental problem is that of learning, through interaction, the best action (that is the action which is rewarded optimally) allowed by the environment. By using running estimates of reward probabilities to learn the optimal action, an extremely efficient Pursuit Algorithm was earlier reported [24, 26, 27, 28] which is presently among the fastest algorithms known. This paper investigates the improvements gained by rendering the Pursuit Algorithm discrete, and this is done by restricting the probability of selecting an action to a finite, and hence, discrete subset of [0, 1]. This improved scheme is proven to be **optimal in probability** (implying ε-optimality) in all stationary environments. Furthermore, our experimental results seem to indicate that the algorithm presented in this paper is the fastest absorbing learning automaton reported in the literature to date. Comparison with the continuous form of the pursuit algorithm are also presented.

* Partially supported by the Natural Sciences and Engineering Research Council of Canada. A preliminary version of this paper will be presented as an invited paper at the 1989 IEEE International Conference on Systems, Man and Cybernetics, November 1989 in Boston, Mass.

I. INTRODUCTION

A primary goal of artificial intelligence is that of extending the use of computers to solve problems typically handled by the human intellect. Rather than requiring a complete database with which to solve a particular class of problems, ideally, the intention is that the automaton should be capable of arriving at the best solution with the information available. Tsetlin created a model of computer learning called the learning automata (LA) as an archetype for a possible solution to this problem. Tsetlin modelled a biological type of learning [20], where the organism is born with relatively little initial knowledge and learns which actions are appropriate through trial and error. While interacting with the environment, the LA selects one from a set of actions, and feedback from the environment tells the LA if the chosen action was rewarded or penalized. The LA then uses this information to decide which action to take next, and the cycle continues. In the study of LA the environment is typically modelled as one which rewards or penalizes the machine randomly; the LA attempts to find which strategy is optimal, that is, maximises the probability of being rewarded. The key point is that the probability of success for each action is unknown to the automaton; it adapts itself to the environment by learning the optimal action.

Learning algorithms are useful whenever complete knowledge about an environment is unknown, expensive to obtain or impossible to quantify. Thus they have found applications in various fields including game playing [1,5,7], pattern recognition [11, 23], hypothesis testing [11] and object partitioning [21,22]. Learning automata are also useful when the environment in which they interact are time varying and thus have found applications in priority assignments in a queuing system [9], and the routing of telephone calls [12, 13].

The varieties of learning automata have been reviewed by Lakshmivarahan [5], and by Narendra and Thathachar [10, 11], and so, in this paper only a rough classification will be outlined. Initially the LA designed were of a fixed structure in the sense that the decision functions used by the automaton were time invariant. Notable examples of this type were designed by Tsetlin, Krylov, and Krinsky [29, 30]. Later, variable structure stochastic automata (VSSA) were developed in which the decision process was generalised so that the state transition probabilities and the action selecting probabilities evolved with time. In fact, a VSSA is completely characterised by the method of updating the probability of choosing an action [5,8,10,11].

The probability that an action may be successful can remain fixed (termed stationary) or it can vary with time (termed non-stationary) depending on the environment; VSSA have been developed that are suitable for each situation. In general, the VSSA are analysed from the point of view of the underlying Markovian processes which they represent. Thus families of VSSA which possess absorbing barriers have been studied[8, 11, 14, 19]. An absorbing barrier is a state that has the property that if the automaton enters this state it is locked there for the rest of time.

Ergordic VSSA have also be investigated [11,12,17,18,19,29]. These VSSA converge in distribution and thus the asymptotic distribution of the action probability vector has a value which is independent of the corresponding initial vector. In non-stationary environments, if the optimal action changes with time, an ergodic VSSA can follow it; in stationary environments, automata with absorbing barriers may be preferred because they can be made converge to the optimal action with a probability as close to unity as desired.

In practice, one of the limiting factors in the applications involving LA is their slow rate of convergence. A general approach for improving many VSSA has been proposed in [25]; this is done by rendering the probability of choosing an action discrete. Such automata are the discretized versions of their continuous counterparts. The concept of discretization is implemented by restricting the probability of choosing an action to only finitely many values from the interval [0, 1]. The discretization is said to be linear if the values allowed are equally spaced in the interval [0, 1] and includes the points 0 and 1; otherwise, the discretization is called non-linear.

The motivation for this modification is illustrated in Figure 1. Once the optimal action has been determined, say at time t, and the corresponding action probability is relatively close to unity, the discretized automaton increases the probability of choosing that action to the value of unity **directly**, rather than approach the value unity asymptotically. Figure 1 illustrates that for the continuous case A, if the probability of choosing the optimal action is at 0.96, and the automaton is rewarded five times, then at time t + 5 the probability of choosing the optimal action will be close to 1.0. In fact, the closeness will depend on the parameters of the scheme. However in the discretized case, B, the probability of choosing the optimal action will be exactly unity if the probability space is divided into intervals of width 0.01. Thus the rate of convergence is frequently improved.

Another benefit of discretizing the probability of choosing an action is that it minimises the requirements on the system's random number generators (RNG) [16,19]. This is important since VSSA use RNG [21] in their implementation. In theory, it is assumed that any real value in [0, 1] can be obtained; in practice, only a finite number of these values are available. This is due to the algorithm used for generating random numbers and the fact that the values are truncated after a fixed number of decimal places due to the finite precision allowed by the (finite memory) representations of floating point numbers. Experiments involving discretizing were first described by Thathachar and Oommen [25] and theoretical results proved in [15, 16, 19] showed the improvements gained when various automata were discretized.

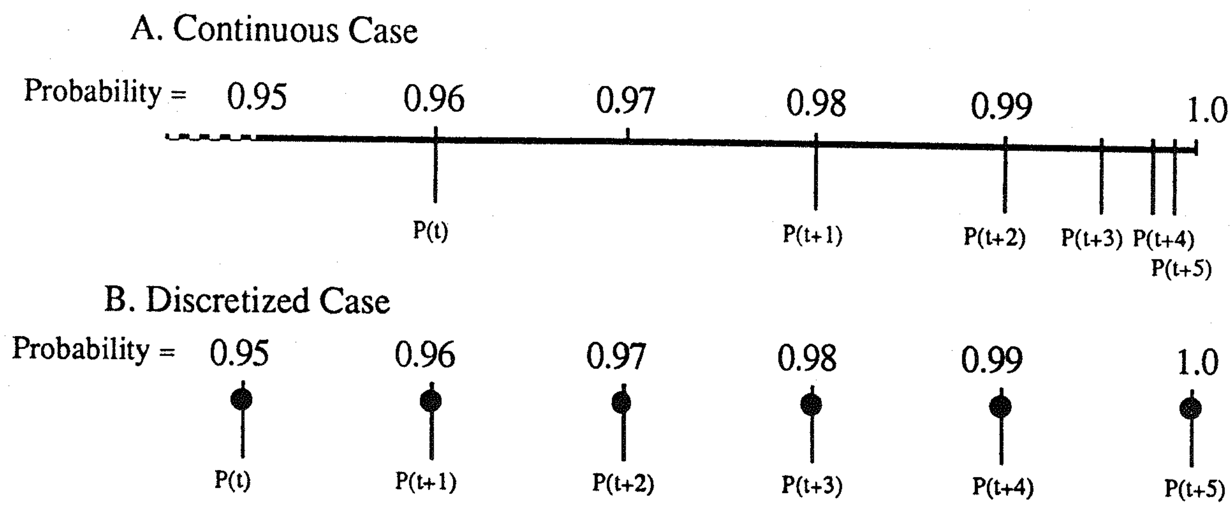

Figure 1: A Comparison of the Probability of Choosing the Optimal Action for a Continuous and a Discretized Learning Algorithm

Another approach to improving the rate of convergence of learning automata was introduced by Thathachar and Sastry with the so-called estimator algorithms [24, 26, 27, 28]. Typically, non-estimator algorithms update the probability of choosing an action based directly on the feedback obtained from the environment. If an action was rewarded then the probability of choosing that action at the next time instant would be increased. These algorithms are characterised by the use of a running estimate of the probability of reward for each action. When an action is selected and feedback from the environment is obtained, the estimator algorithms update the estimate of the probability of reward for that action. The change in the probability of choosing an action is then based on the running estimates of the probability of reward rather than on the feedback from the environment alone. So even if a particular action was rewarded, it may happen that the probability of choosing another action is increased. Pursuit Algorithms are a subclass of estimator algorithms. These algorithms are characterised by the fact that they pursue the optimal action in the sense that at every time instant, they increase the probability of selecting the action whose current estimate of being rewarded is maximal [27]. The computational superiority of the pursuit algorithms and the traditional algorithms have been well proven in [24, 26-28].

In this paper we shall design a new family of schemes obtained by discretizing a certain family of **absorbing** pursuit algorithms. The fact that the continuous pursuit algorithms are an order of magnitude faster than the L_{RI} scheme was convincingly demonstrated elsewhere [24, 28]. However, by discretizing the probability space we have been able to obtain a scheme which possesses properties similar to those possessed by the continuous algorithm. Additionally, it also possesses all the advantages of a discretized scheme. As a result of the experiments we have conducted we believe that for a given accuracy, the DPL_{RI} scheme is faster than its continuous counterpart. Indeed, it is probably the fastest converging **absorbing** automaton reported in the literature. The accuracy of the scheme is remarkable too.

I.1 Fundamentals

An automaton interacting with an environment is shown in Figure 1. The automaton consists of four components denoted by $< A, B, Q, T >$. A is the set of r actions that the automaton can choose from. The various actions are then elements of $A = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$. The automaton is allowed one choice at each time instant t and its choice is denoted as $\alpha(t)$. This choice is constrained by the requirement that $\alpha(t) \in A$, for all t. The set of responses from the environment is denoted by B, where B is the set $\{0, 1\}$. $\beta(t)$ is the response from the environment at time t; thus $\beta(t) \in B$ for all t. If at an instant of time t, $\beta(t) = 1$, then this means the automaton is rewarded for action $\alpha(t)$. Similarly, $\beta(t) = 0$ if the automaton is penalised for action $\alpha(t)$. The reader should note that this notation is not used universally; often in the literature $\beta(t) = 1$ if the automaton is penalised.

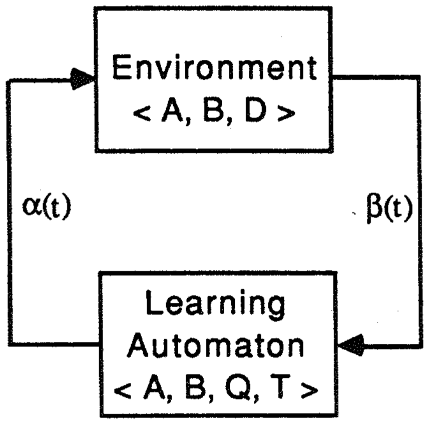

Figure 2: A Learning Automaton Interacting with the Environment

The probability that the environment will reward action α_i at time t is denoted by $d_i(t)$. It is assumed that the environment is stationary. This means that $d_i(t)$ is constant for all time t and so the index of time is dropped and the quantity is denoted as d_i. The set $\{d_1, d_2, \ldots, d_r\}$ is denoted as D. The cardinality of D is r because there is one d_i for each action α_i. It is also assumed that there is a unique maximum for the set D called d_b, where $d_b = \mathbf{Max}_{1 \le i \le r}\{d_i\}$. The action possessing this probability of reward, namely α_b, is referred to as the best action. The value of each d_i is unknown to the automaton, and so its task is to decide which action is the best. It bases its decision on the information gained by selecting actions, and seeing if the choice is rewarded. This cycle continues until the learning process is terminated.

The vector state of the automaton is denoted by Q. The state consists of two r-tuples namely **P** and **D'**. When Q or any of its components are indexed by t, this represents the vector state of the automaton at time t. The i th component of **P**, denoted $p_i(t)$ is the probability that the i th action will be chosen at time instant t. For the continuous versions of the Pursuit Algorithm the value of $p_i(t)$ can be any real number between 0 and 1. In the discrete version, there are only finitely many values for $p_i(t)$, namely $p_i(t)$ is one of $\{0, \Delta, 2\Delta, 3\Delta, \ldots, 1\}$, for all t. Here Δ is referred to as the smallest step size and is a direct function of the total number of subdivisions of the probability space [0,1]. This parameter Δ is dependent on two quantities, the number of actions r, and a resolution parameter, n; it is defined by $\Delta = 1/rn$. The probability of selecting an action is initially set at $1/r$ for all actions. Normally this is the only state vector that the learning automaton includes; however, the estimator class of algorithms expands their state to include another quantity d'_i. This additional parameter represents the ongoing estimate at time t of d_i, the probability of reward if the ith action is taken. To maintain the latter, $d'_i(t)$ is calculated as the number of times the ith action has been rewarded up to time t (denoted by $W_i(t)$), divided by the number of times the ith action has been selected up to time t (denoted by $Z_i(t)$).

The learning algorithm, denoted as T, adjusts Q(t). The general structure of the learning algorithms considered here are the same. An action is selected based on the existing probability distribution. The selected action is presented to the environment, and the automaton is updated based on the response from the environment. After considering the various learning criteria we will describe both continuous and discretized versions of the Pursuit Algorithm.

I.2 Learning Criteria

The probability that the automaton will pick the best action is denoted p_b.

A learning automaton is optimal if $p_b(t) \to 1$ as $t \to \infty$, with probability 1.

A learning automaton is said to be ε-optimal if the probability of choosing the best action can be made as close to unity as desired. More explicitly, it is said to be ε-optimal if for all $\varepsilon > 0$,

$$\liminf_{t \to \infty} p_b(t) > 1 - \varepsilon \qquad (1)$$

Alternatively it can be shown that if there is an internal parameter n such that for all $\varepsilon > 0$, $\delta > 0$, there exists $n_o > 0$, and a $t_o < \infty$ such that $\Pr[\,|\,p_b(t) - 1\,| < \varepsilon\,] > 1 - \delta$ for all $t \geq t_o$ and for all $n > n_o$ then the scheme is ε-optimal [5]. Finally, a learning automata is said to be optimal in probability if $p_b(t) \to 1$ in probability as $t \to \infty$. Optimality in probability implies ε-optimality.

II. CONTINUOUS PURSUIT ALGORITHMS

II.1 Pursuit Algorithm 1

The continuous Pursuit Algorithm considered here [28] adjusts $Q(t)$ in three steps. The first step is to pick the action $\alpha(t)$ based on probability distribution $\mathbf{P}(t)$.The second step is to update $\mathbf{P}(t)$ based on feedback from the environment. Once an action is rewarded, then every action probability is reduced by a factor of $1-\lambda$, where λ is the parameter of the scheme, and is a quantity close to zero. Subsequently, the action probability that has the highest running estimate of being rewarded is increased by λ. However, if the action was penalised then the probabilities stay the same. Finally, the third step is to update the running estimates for the probability of being rewarded. To do this two more variables are introduced. Z_i is the number of times the ith action has been chosen up to time t and W_i is the number of times the ith action has been rewarded up to time t. Observe that for all j, the estimate $d'_j(t)$ equals W_j / Z_j. Explicitly the algorithm is stated as follows :

ALGORITHM Pursuit Algorithm 1

Step 1: At time t pick $\alpha(t)$ according to probability distribution $\mathbf{P}(t)$.

Step 2: Update $\mathbf{P}(t)$ according to the following:

$$\begin{aligned} \mathbf{P}(t+1) &= \mathbf{P}(t) + \lambda(\mathbf{e}_m - \mathbf{P}(t)) && \text{if } \beta(t) = 1 \\ &= \mathbf{P}(t) && \text{if } \beta(t) = 0 \end{aligned} \tag{2}$$

where m satisfies $d'_m(t) = \mathbf{Max}_i\{ d'_i(t) \}$,
$0 < \lambda < 1$, λ is the parameter of the scheme, and,
where $\mathbf{e}_m$ is the unit r-vector with 1 in the mth co-ordinate.

Step 3: Update $\mathbf{D}'(t)$ according to the following:

$$\begin{aligned} \text{if } \alpha(t) = \alpha_j, \quad & W_j(t+1) = W_j(t) + \beta(t) \\ & Z_j(t+1) = Z_j(t) + 1 \\ & d'_j(t+1) = W_j(t+1) / Z_j(t+1) \\ \text{for all } i \neq j \quad & W_i(t+1) = W_i(t) \\ & Z_i(t+1) = Z_i(t) \\ & d'_i(t+1) = d'_i(t). \end{aligned}$$

END ALGORITHM Pursuit Algorithm 1

II.2 Pursuit Algorithm 2

The second pursuit algorithm described in the literature adjusts Q(t) in three steps as well. This algorithm is formally described as below.

ALGORITHM Pursuit Algorithm 2

Step 1:Same as Pursuit Algorithm 1.

Step 2: Let $\alpha(t) = \alpha_j$, update **P**(t) according to the following:

$$p_i(t+1) = p_i(t) + \lambda \sum_{j \neq i} \left[f(d'_i(t)) - f(d'_j(t)) \right] \left[S_{ij}(t) p_j(t) + S_{ji}(t) \frac{p_i(t)}{r-1} (1 - p_j(t)) \right]$$

$$p_j(t+1) = p_j(t) - \lambda \quad \left[f(d'_i(t)) - f(d'_j(t)) \right] \left[S_{ij}(t) p_j(t) + S_{ji}(t) \frac{p_i(t)}{r-1} (1 - p_j(t)) \right] \quad j \neq i \qquad (3)$$

where $0 < \lambda < 1$, λ is the parameter of the scheme, and
where $S_{ij}(t)$ is an indicator function such that $S_{ij}(t) = 1$ if $d'_i(t) > d'_j(t)$
$= 0$ if $d'_i(t) \leq d'_j(t)$.
Here f is any monotonically increasing function $f:[0,1] \rightarrow [0,1]$.

Step 3: Same as Pursuit Algorithm 1.

END ALGORITHM Pursuit Algorithm 2

Note that **P**(t+1) depends indirectly on the response from the environment. The value of **D**'(t) affects the sign of $f(d'_i(t)) - f(d'_j(t))$ and $S_{ij}(t)$. This means that if α_i is chosen and $d'_i(t) < d'_j(t)$ then the value of $d'_j(t)$ is increased proportional to $p_i(1-p_j)/(r-1)$. If, on the other hand, $d'_i(t) > d'_j(t)$ then the value of $d'_j(t)$ is decreased proportional to p_j. The $S_{ij}(t)$ term in step 2 determines which of the two factors $p_i(1 - p_j) / (r-1)$ or p_j multiply the sum.

It is easy to see that whereas the Pursuit Algorithm 1 is an **absorbing** algorithm, the latter scheme is ergodic. In principle, the probability of selecting the best action approaches unity asymptotically but due to the finite precision of floating point numbers as represented by the machine, this proabability will eventually round off to one. Once this has happened no combination of penalities or rewards can change or reverse the situation. Pursuit Algorithm 2 is essentially of an ergodic form. If the probability of choosing a particular action becomes unity, it can be reduced if the action is penalized a sufficient number of times. In this paper, since we are dealing with an absorbing discretized pursuit algorithm, we shall not discuss the latter algorithm in any further detail. We shall now present a discretized version of Pursuit Algorithm 1 and shall prove its asymptotic properties. Subsequently, by presenting simulation results, we shall show that the discretized version of the algorithm is much faster than the corresponding continuous algorithm. We are currently working on discretizing Pursuit Algorithm 2 and proving its analytical properties. We believe that analogous comparisons apply between the continuous and the discretized versions of Pursuit Algorithm 2 too.

III. DISCRETIZED PURSUIT LINEAR REWARD-INACTION LEARNING AUTOMATA-DPL$_{RI}$

The primary goal of the Discretized Pursuit Linear Reward-Inaction Learning Automaton (DPL$_{RI}$) is to mimic the learning process followed by the continuous Pursuit Algorithm 1, except that the probability changes are made in discrete steps. Thus the algorithm works in three steps just like its continuous counterpart, the Pursuit Algorithm 1. The difference is in the second step where the components of the probability vector are increased by integral multiples of the smallest step size Δ, where $\Delta = 1/rn$, r being the number of actions and n being a parameter called the resolution parameter. If the automaton has been rewarded and has not converged, then all the action probabilities are decreased by Δ except the one with the highest estimate of the reward probability. This action probability is increased by the appropriate amount to keep the vector as a probability.

Note that the above algorithm can be implemented using primarily integer counter updates. This is because the action probability for any action α_i need not be stored as a real number but only as an integer k_i, whence the action probability can be computed as $k_i.\Delta$. Explicitly, the algorithm is as described below.

ALGORITHM Discretized Pursuit L$_{RI}$

Step 1: Same as Pursuit Algorithm 1.

Step 2: Let $\alpha(t) = \alpha_k$, update **P**(t) based on feedback from the environment.

Let m be the index of the action whose estimate of reward probability is maximum at time t. Explicitly, m satisfies $d'_m(t) = \mathbf{Max}_{1 \le i \le r}\{ d'_i(t) \}$. Then, if $\Delta = 1/rn$,

$$\begin{aligned} p_j(t+1) &= \mathbf{Max}(p_j(t) - \Delta, 0) && \text{for all } j \neq m, \text{ if } \beta(t) = 1 \text{ and } p_m(t) \neq 1 \\ p_m(t+1) &= 1 - \textstyle\sum_{j \neq m} \mathbf{Max}(p_j(t) - \Delta, 0) && \text{if } \beta(t) = 1. \\ p_j(t+1) &= p_j(t) && \text{for all } 1 \le j \le r \text{ otherwise.} \qquad (5) \end{aligned}$$

Step 3: Same as Pursuit Algorithm 1.

END ALGORITHM Discretized Pursuit L$_{RI}$

Note that the condition $p_m(t) \neq 1$ in Step 2 is superfluous. It has been included for the ease of understanding future expressions.

We proceed to examine the theoretical properties of the DPL_{RI} algorithm. The proof of the results follow the fundamental streamlined concepts used in the proofs of Pursuit Algorithms 1 and 2 [24, 26-28]. However, there are two differences to note. The first is trivial. The continuous pursuit algorithms have a learning parameter λ and as λ tends to zero the algorithm increases in its accuracy and simultaneously takes longer to converge. The discretized version has the parameter n, called the resolution parameter, which is bounded by one and infinity. As n increases the algorithm increases in its accuracy and simultaneously takes longer to converge. Furthermore, whereas the parameter of the continuous schemes vary continuously, the parameter of the discrete scheme varies in a discrete manner. Thus the limiting arguments have to be used in somewhat different ways.

IV. PROOF OF CONVERGENCE

We shall now prove the convergence of the scheme. Indeed we will show that in every stationary random environment, the Discretized Pursuit Linear Reward-Inaction algorithm (DPL_{RI}) is ε-optimal. This will be done by showing that given $\varepsilon > 0$ and $\delta > 0$, there exists an $n_o > 0$ and a $t_o < \infty$ such that for all time $t \geq t_o$ and for any resolution parameter $n > n_o$ the following is true:

$$\mathbf{Pr}[\,|\,p_b(t) - 1\,| < \varepsilon] > 1 - \delta.$$

Note that p_b will refer to the probability of choosing the best action. We will proceed by way of a few lemmas. Our first lemma derives a lower bound for the probability of selecting the ith action subject to the DPL_{RI} algorithm.

Lemma 1

Let $\mathbf{P}(t) = [p_1(t), p_2(t),\ldots, p_r(t)]^T$, where $p_i(t)$ is the probability that the DPL_{RI} automaton selects action α_i at time t. Then the updating function of the DPL_{RI} guarantees that $p_i(t) \geq (n - t)/nr$ where n is the resolution parameter and r is the number of actions.

Proof :

This proof is achieved as a simple induction on t.

Basis Step

Consider the case when t = 0. By the initialisation process we know that

$$p_i(0) = 1 / r \quad \text{for all i.}$$

Clearly this satisfies the theorem since $p_i(0) = (n-0) / (rn)$ for all i.

Inductive Step

Assume that $p_i(t) \geq (n - t) / (rn)$.

Then using (5) we have $p_i(t+1) \geq p_i(t) - 1/(rn)$, and by induction hypothesis we have

$$\begin{aligned} p_i(t+1) &\geq (n - t) / (rn) - 1/(rn) \\ &= (n - 1 - t) / (rn) \\ &= (n - (t+1)) / (rn) \end{aligned}$$

and the result is proved. ●●●

The next lemma states a property concerning the number of times any arbitrary action has been chosen.

Lemma 2

Let the random variable Y_t^i represent the number of times the i th action was chosen up to time t. Then **Var**$[Y_t^i]$ is always bounded from above by **E**$[Y_t^i]$.

Proof :

We begin by first defining a random variable X_t^i as the indicator function which has the value 1 if the i th action is chosen at the fixed time t, and 0 otherwise. Explicitly,

$$\begin{aligned} X_t^i &= 1 \text{ if } \alpha(t) = \alpha_i \\ &= 0 \text{ if } \alpha(t) \neq \alpha_i \end{aligned} \qquad (6)$$

$$\text{and hence } Y_t^i = \sum_{k=1}^{t} X_k^i$$

It can easily be seen by direct calculation that since X_t^i is Bernoulli distributed, the following results are true:

$$E[X_t^i] = p_i(t) \quad \text{and} \quad \mathbf{Var}[X_t^i] = p_i(t) - p_i(t)^2 \qquad (7)$$

Now we will show that there is no correlation between X_t^i and X_s^i if $s \neq t$. This is achieved by proving that

$$E[X_t^i]\,E[X_s^i] = E[X_t^i X_s^i]. \qquad (8)$$

First consider the left hand side of (8). By virtue of (6):

$$E[X^i_t]\,E[X^i_s] = p_i(t)\;p_i(s)$$

Now consider the right hand side of (8). The second joint moment is:

$$E[X^i_t X^i_s] = \Sigma_{j,k}\; j\cdot k\cdot \mathbf{Pr}\{\,(X^i_t = j)\,(X^i_s = k)\,\}$$

$$= 1\cdot p_i(t)p_i(s) + 0\cdot\{\,p_i(s)(1-p_i(t)) + p_i(t)(1-p_i(s)) + (1-p_i(t))(1-p_i(s))\,\}$$

$$= p_i(t)\,p_i(s)$$

Thus the left hand side of (8) equals the right hand side of (8) and there is no correlation between X^i_t and X^i_s if $s \neq t$. Using the laws of elementary probability this leads to the following:

$$E[Y^i_t] = \Sigma_{k=1}^{t}\; E[X^i_k]$$

$$\mathbf{Var}[Y^i_t] = \Sigma_{k=1}^{t}\; \mathbf{Var}[X^i_k] \qquad (9)$$

With property (7) and (9) the lemma follows in a straight forward manner. Indeed, by (7),

$$\mathbf{Var}[Y^i_t] = \Sigma_{k=1}^{t}\; \mathbf{Var}[X^i_k]$$

$$= \Sigma_{k=1}^{t}\, p_i(k) \;-\; \Sigma_{k=1}^{t}\, p_i(k)^2$$

$$\leq \Sigma_{k=1}^{t}\, p_i(k)$$

$$= \mathbf{E}[Y^i_t],$$

and hence $\mathbf{Var}[Y^i_t]$ is bounded from above by $\mathbf{E}[Y^i_t]$. ●●●

The series of lemmas will be completed by showing that the expected value of the number of times that an action has been chosen possesses a well defined lower bound.

Lemma 3

For any positive integer t there exists a value for the resolution parameter, n, for the DPL_{RI} scheme, such that $E[Y^i_t] \geq t\,p_i(0)/2$.

Proof :

Consider $E[Y^i_t]$. By (6) and (7) we have for all $t < n$,

$$E[Y^i_t] = \Sigma_{k=1}^{t}\, p_i(k)$$

$$\geq \Sigma_{k=1}^{t}\; (n-k)/rn$$

the latter being a consequence of Lemma 1. By expanding and computing the sum using the closed form for the sum of a geometric series we have:

$$E[Y_t^i] = t/r - (1/rn)\sum_{k=1}^{t} k$$
$$= t/r - t(t+1)/(2rn)$$

Since t and n are integers,the inequality $n > t$ implies $n \geq t + 1$, which in turn implies that $1/n \leq 1/(t+1)$, and hence $-1/n \geq -1/(t+1)$. Thus the right hand side of the above equality leads to the following inequality:

$$E[Y_t^i] \geq t/r - t(t+1)/(2r(t+1))$$
$$= t/r - t/2r$$
$$= t/2r$$

Since $1/r$ equals the initial probability of selecting an action $p_i(0)$, we have $E[Y_t^i] \geq t\, p_i(0)/2$ and so the result is proved. ●●●

The first and second moments of Y_t^i will be used with Chebyshev's Inequality which states that if Y_t^i is a random variable with finite mean μ and variance σ^2, then for any value $k > 0$,

$$\Pr\{\, |Y_t^i - \mu| > k \,\} \leq \sigma^2 / k^2.$$

Observe that if we set $k = E[Y_t^i] - M$, we now have,

$$\Pr\{\, |Y_t^i - E[Y_t^i]| > E[Y_t^i] - M \,\} \leq \sigma^2 / (E[Y_t^i] - M)^2 \qquad (10)$$

Thus if σ^2 increases slower than $E[Y_t^i] - M$, then as t gets larger, this inequality can be used to reduce the probability to a quantity to a value which is as small as is desirable. This concept is now used in the following theorem.

Theorem 1

For any given constants $\delta > 0$ and $M < \infty$, there exists $n_o > 0$ and $t_o < \infty$ such that under the Discretized Pursuit Linear Reward-Inaction learning algorithm, for all learning parameters $n > n_o$ and all time $t > t_o$: $\Pr\{$each action chosen at least M times at time t$\} \geq 1 - \delta$.

Proof :

Define the random variable Y_t^i as in Lemma 2. Then we shall show that $\Pr\{Y_t^i > M\} \geq 1 - \delta$. Using elementary probability theory we know that for all events A and B,

$$\text{if } A \Rightarrow B \text{ then } \Pr\{A\} \leq \Pr\{B\} \qquad (11)$$

Let A be the event : $Y_t^i < M$, and,

let B be the event: $| Y_t^i - E[Y_t^i] | \geq E[Y_t^i] - M$.

We shall show that $A \Rightarrow B$. To do this consider the L.H.S. of B as

$$\begin{aligned} | Y_t^i - E[Y_t^i] | &= | E[Y_t^i] - Y_t^i | \\ &\geq | E[Y_t^i] | - | Y_t^i | \end{aligned}$$

the latter being true due to the trianglular law of inequalities. Since A is assumed true this yields:

$$\begin{aligned} | Y_t^i - E[Y_t^i] | &\geq | E[Y_t^i] | - | M | \\ &= E[Y_t^i] - M, \end{aligned}$$

the last equation following because the quantities are positive. Combining the latter inequality with (11) we have,

$$\mathbf{Pr}\{ Y_t^i < M \} \leq \mathbf{Pr}\{ | Y_t^i - E[Y_t^i] | \geq E[Y_t^i] - M \}$$

This implies that for all t, there exists a value of M such that $E[Y_t^i] > M$.

We shall now show that:

$$\mathbf{Pr}\{ Y_t^i < M \} \leq \mathbf{Pr}\{ | Y_t^i - E[Y_t^i] | \geq E[Y_t^i] - M \}$$

Using Chebyshev's Inequality in the form of (10) we have,

$$\begin{aligned} \mathbf{Pr}\{ Y_t^i < M \} &\leq \mathrm{Var}[Y_t^i] / (E[Y_t^i] - M)^2 \\ &\leq E[Y_t^i] / (E[Y_t^i] - M)^2 \qquad \text{(by Lemma 2),} \\ &= E[Y_t^i] / (E[Y_t^i]^2 - 2M\, E[Y_t^i] + M^2) \\ &= 1 / (E[Y_t^i] - 2M + M^2 / E[Y_t^i]). \end{aligned}$$

Since, by Lemma 3 $E[Y_t^i]$ can be arbitrarily large, therefore $\mathbf{Pr}\{ Y_t^i < M \}$ can be made correspondingly as small as is desired. Thus for any action α_i the inequality can be made smaller than δ by having $t > t(i)$ and $n > n_{t(i)}$.

Since, we can repeat this argument for all the actions, we can define t_o and n_o as follows: $t_o = \mathbf{Max}_{1\leq i\leq r}\{ t(i) \}$ and $n_o = \mathbf{Max}_{1\leq i\leq r}\{ n_{t(i)} \}$. Thus for all i we have for all $t > t_o$ and all $n > n_o$, $\mathbf{Pr}\{ Y_t^i < M \} < \delta$ and the theorem is proved. ●●●

We shall now prove that if the mth action is rewarded the most from time t_o onward then the action probability vector for the DPL_{RI} scheme will converge to a unit vector in which the probability of choosing α_m is unity and the probability of choosing any other action is zero.

Theorem 2

Suppose there exists an index m and a time instant $t_o < \infty$ such that $d'_m(t) > d'_j(t)$ for all j such that $j \neq m$ and all $t \geq t_o$, then there exists an integer n_o such that for all resolution parameters $n > n_o$, $p_m(t) \to 1$ with probability one as $t \to \infty$.

Proof :

Consider the submartingale defined by the sequence of random variables $\{p_m(t)\}_{t\geq 0}$ satisfying $\sup_{t\geq 0} \mathbf{E}[| p_m(t) |] < \infty$. It is known [4], that under these conditions there exists a random variable to which $\{ p_m(t) \}$ converges with probability one, i.e.

$$\mathbf{Pr} \{\lim_{t \to \infty} p_m(t) = p_m^{\infty} \} = 1$$

Using the DLP_{RI} algorithm, we know that if m satisfies

$$d'_m = \mathbf{Max}_i \{ d'_i(k) \}$$

then, $d'_m(t) > d'_j(t)$ for all $j\neq m$ and all $t \geq t_o$. Therefore, for all $t > t_o$,

$$\begin{aligned} p_m(t+1) &= 1 - \Sigma_{j\neq m} \mathbf{Max}(p_j(t) - \Delta, 0) && \text{if } \beta(t) = 1 \text{ with probability } d_m \\ &= p_m(t) && \text{if } \beta(t) = 0 \text{ with probability } 1- d_m \end{aligned}$$

If $p_m(t) = 1$ then the absorbing property of the algorithm trivially proves the result.

Of course there is the possibility that the algorithm has already converged to an action α_j where $j\neq m$. To ensure that this has not happened we assume that the resolution parameter is large enough so that the algorithm has not converged by the time $d'_m(t) > d'_j(t)$. Assuming the algorithm has not converged, there exists at least one non-zero ccomponent of $\mathbf{P}(t)$ say $p_k(t)$, and hence we assert that :

$$\mathbf{Max}\ (p_k(t) - \Delta, 0) < p_k(t).$$

Since $\mathbf{P}(t)$ is a probability vector $p_m(t) = 1 - \Sigma_{j\neq m} p_j(t)$, and so,

$$1 - \Sigma_{j\neq m} \mathbf{Max}(p_j(t) - \Delta, 0) > p_m(t).$$

As long as there is at least one non-zero component $p_k(t)$ (where $k\neq m$), it is clear that we can decrement $p_k(t)$ and hence increment $p_m(t)$ by at least Δ. Hence,

$$p_m(t+1) = p_m(t) + c_t\Delta,$$

where is $c_t\Delta$ is an integral multiple of Δ, c_t is bounded by 0 and r, and Δ is the smallest step size. Therefore we write,

$$\begin{aligned} E[\, p_m(t+1) \mid Q(t), p_m(t) \neq 1\,] &= d_m \{ p_m(t) + c_t\Delta \} + (1 - d_m) \{ p_m(t) \} \\ &= p_m(t) + d_m c_t \Delta \end{aligned}$$

Since all the above terms have an upper bound of unity, $E[\, p_m(t+1) \mid Q(t), p_m(t) \neq 1]$ is bounded. Hence $\sup_{t\geq 0} E[\, | p_m(t+1) | Q(t), p_m(t) \neq 1] < \infty$. Thus,

$$E[\, p_m(t+1) - p_m(t) \mid Q(t)\,] = d_m c_t \Delta \geq 0 \text{ or all } t \geq t_o,$$

implying that $p_m(t)$ is a submartingale. By the submartingale convergence theorem [4], $\{p_m(t)\}$ converges and so as $t \to \infty$,

$$E[\, p_m(t+1) - p_m(t) \mid Q(t)\,] \to 0 \text{ w.p. } 1$$

impliying that $d_m c_t \Delta \to 0$ w.p. 1. This in turn implies that $c_t \to 0$ w.p. 1, and consequently that $\sum_{j\neq m} \mathrm{Max}(\, p_j(t) - \Delta, 0) \to 0$ w.p. 1. Hence $p_m(t) \to 1$ w.p. 1, and the theorem is proved.

●●●

Theorem 3

In every stationary random environment, the DPL_{RI} algorithm is ε-optimal. In other words, given $\varepsilon > 0$ and $\delta > 0$, there exists a $n_o > 0$ and a $t_o < \infty$ such that for all $t \geq t_o$ and $n > n_o$:

$$\Pr[\, | P_m(t) - 1 | < \varepsilon] > 1 - \delta$$

Proof :

Let h be the difference between the two highest reward probabilities. By assumption d_b is unique so there exists and $h > 0$, such that $d_b - h \geq d_i$ for all $i \neq m$. By the weak law of large numbers we know that if $\{X_t\}$ is a sequence of independent and identically distributed random variables, each having finite mean, μ, then for any $\varepsilon > 0$,

$$\Pr\{\, | (X_1 + X_2 + \ldots + X_t)/t - \mu | > \varepsilon \} \to 0 \text{ as } t \to \infty.$$

By defining each X_t as X_t^i as in (6) and using the above law we have that for a given $\delta > 0$ there exists an $M_i < \infty$, such that if α_i is chosen and least M_i times:

$$\mathbf{Pr}\{ \mid d'_i(t) - d_i \mid < h / 2\} > 1 - \delta.$$

Let $M = \mathbf{Max}_{1\le i\le r}\{M_i\}$. Since h is strictly positive, it is clear that for all $j \neq m$ and for all t, if $\mathbf{Min}_{1\le i\le r}\{ Y_t^i \} > M$ then the $\mathbf{Pr}\{ |d'_m(t) - d_j| < h/2\} > 1 - \delta$.

By Theorem 1 we know that we can define t_o and n_o such that for all $t > t_o$ and all $n > n_o$

$$\mathbf{Pr}\{ \mathbf{Min}_{1\le i\le r}\{Y_t^i\} > M \} > 1 - \delta.$$

Thus if all actions are chosen at least M times then each of the d'_i will be in an $h/2$ neighbourhood of d_i. Hence we have,

$$\begin{array}{lll} d_m - h/2 & > d_m - h & \text{because } h > 0 \\ & \ge d_i & \text{for all } i \neq m. \end{array}$$

Since M is a constant for each environment, by Theorem 1 there exists a time instant $t_o < \infty$ and $n_o > 0$ such that under the DPL_{RI}, for all $n > n_o$ and $t > t_o$,

$$\mathbf{Pr}\{ Y_t^i > M \} \ge 1 - \delta. \qquad (12)$$

By the law of total probability $\mathbf{Pr}\{A\} = \mathbf{Pr}\{A \mid B\}\, \mathbf{Pr}\{B\} + \mathbf{Pr}\{A \mid B^c\}\,(1 - \mathbf{Pr}\{B\})$

Since probability is a continuous set function we have:

$$\lim_{t\to\infty}\mathbf{Pr}\{A\} = \lim_{t\to\infty}\mathbf{Pr}\{A|B\} \lim_{t\to\infty}\mathbf{Pr}\{B\} + \lim_{t\to\infty}\mathbf{Pr}\{A \mid B^c\}\,(1 - \lim_{t\to\infty}\mathbf{Pr}\{B\}).$$

Let A and B be the following events:

$$A \equiv | p_m - 1 | < \varepsilon, \text{ and,}$$

$$B \equiv \mathbf{Max}_i \{| d'_i(t) - d_i |\} < h / 2.$$

Then clearly,

$$\mathbf{Pr}\{A \mid B\} = \mathbf{Pr}\{ \mid p_m - 1 \mid < \varepsilon \,\Big|\, \mathbf{Max}_i \{| d'_i(t) - d_i | \} < h/2 \}.$$

Now using Thereoms 1 and 2, we know that

$$\lim_{t\to\infty} \mathbf{Pr}\{ A \mid B\} \to 1$$

primarily because we can select a resolution parameter large enough to satisfy both the theorems. Furthermore, we know by Theorem 1 and (12) that

$$\lim_{t\to\infty} \mathbf{Pr}\{B\} \to 1-\delta.$$

Since $\lim_{t\to\infty} \mathbf{Pr}\ \{A \mid B\} = 1$, the corresponding probability $\lim_{t\to\infty} \mathbf{Pr}\ \{A \mid B^c\}$ is exactly 0, and so,

$$\lim_{t\to\infty} \mathbf{Pr}\ \{A\} = \lim_{t\to\infty} \mathbf{Pr}\ \{A \mid B\} \lim_{t\to\infty} \mathbf{Pr}\ \{B\}.$$

Hence $\lim_{t\to\infty} \mathbf{Pr}\ \{ \mid p_m - 1 \mid < \varepsilon \} > 1-\delta$ for all $n > n_0$ and the theorem is proved. ●●●

V. SIMULATION RESULTS

Simulation were performed to compare the rates of convergence of the discrete and continuous versions of Pursuit Algorithm 1. For the continuous version, the algorithm was said to have converged whenever the probability of choosing an action exceeded 0.99. Instead, the discretized version was treated more stoically and said to have converged only when the probability of choosing an action was **exactly unity**. If the automaton converged to the best action (i.e., the one with the highest probability of being rewarded), it was said to have converged accurately.

For both the discretized and the continuous algorithms, the errors for a hundred experiments were calculated by searching through the respective parameter spaces. In each case, the parameters which yielded the **fastest** convergence and which simultaneously resulted in zero errors in one hundred experiments were recorded. Thus in the case of Pursuit Algorithm 1, the **largest** value of λ which yielded a hundred percent accuracy was reported, and in the case of the DPL_{RI}, the **smallest** value of the resolution parameter, n, was the value reported. These parameters were then used to check the rates of convergence of the respective algorithms.

In the first set of simulations the automata were placed in a two action environment. The probability of reward for one action was fixed at 0.8 for all simulations. The probability of reward for the second action was increased form 0.2 to 0.775. Before starting the algorithm estimates for **D'** were obtained by selecting each action ten times as was done by Thathatchar and Sastry[28]. These extra 20 iterations were then included in the total number of iterations until the algorthms converged. The ensemble average results are shown in Table 1.

It can be seen that the DPL_{RI} Algorithm performs better than the continuous Pursuit Algorithms in **all** cases. Indeed, for more difficult tasks the DPL_{RI} converges almost twice as fast as the continuous schemes. Thus, when the environment reward probabilities are (0.8, 0.775), the continuous Pursuit Algorithm required an average of 6190 iterations to arrive at an accuracy of ninty-nine percent. The parameter of the scheme in this case was 0.0008547. The DPL_{RI} Algorithm converged to an accuracy of a hundred percent with an average of 3300 iterations. The computational gain is remarkable.

Note that apart from the computational gain observed in the mean number of iterations, the actual computing effort involved in the DPL_{RI} is significantly less because the probability updates at each iteration are not multiplicative. Indeed, they are effected by mere integer counter updates, because the action probability for any action α_i need not be stored as a real number but only as an integer k_i, whence the action probability can be computed as $k_i\Delta$.

Table 1: Comparison of the Ensemble Average Number of Iterations Required for Convergence for Progressively More Difficult Environments.

Probability Of Reward		Mean Iterations	
Action 1	Action2	Continuous Pursuit Algorithm	Discrete Pursuit Algorithm
0.8	0.2	22	22
0.8	0.4	42	39
0.8	0.6	148	125
0.8	0.7	636	357
0.8	0.75	2980	1290
0.8	0.775	6190	3300
In each case the parameter of the corresponding sheme was chosen so as to yield zero errors in 100 experiments and simultaneously maiximized the rate of convergence.			

For the second set of simulations the algorithms were compared in two ten action environments referred to as Environments A and B respectively. The results are summarized in Table 2. As before, these results include the 100 iterations that where performed to get the initial estimates of the **D**' vector. The rule for convergence was identical to the one used earlier and the value of the resolution parameter was chosen as in the first set of simulations. Note that for Environment A the difference between the two highest reward probabilies is 0.2 and for the Environment B this difference is 0.08. Again the DPL_{RI} always out-performs the Pursuit Algorithm 1. In Environment A the Pursuit Algorithm 1 took an average of 1140 iterations to converge to an accuracy of ninty-nine percent. The parameter of the scheme in this case was 0.0073529. The corresponding number of iterations required by the DPL_{RI} scheme was only 799 obtained with a resolution parameter of 2560. The computational gain obtained by discretizing the probability space is obvious.

Table 2: Comparison of the Ensemble Average Number of Iterations Required for the Convergence of Automata for a Pair of ten Action Environments.

Environment	Mean Iterations	
	Continuous Pursuit Algorithm	Discrete Pursuit Algorithm
A	1140	799
B	2570	1770

In each case the parameter of the corresponding sheme was chosen so as to yield zero errors in 100 experiments and simultaneously maiximized the rate of convergence. The reward probabilities were as follows:

Environment A	0.7	0.5	0.3	0.2	0.4	0.5	0.4	0.3	0.5	0.2
Environment B	0.10	0.45	0.84	0.76	0.20	0.40	0.60	0.70	0.50	0.50

The fact that the continuous pursuit algorithm is an order of magnitude faster than the L_{RI} scheme demonstrated in [28]. As a result of our experiments we observe that for a given accuracy, the DPL_{RI} scheme is faster than its continuous counterpart and is thus probably the fastest converging **absorbing** automaton reported in the literature. Furthermore, the accuracy of the scheme is remarkable too.

VI. CONCLUSIONS

The rate at which learning algorithms converge has been a limiting factor in their implementation. Discretizing provides a general method of improving the preformance of VSSA. Discretizing VSSA restricts the probability of choosing an action to a finite number of values. Discretizing reduces computation time by replacing floating-point multiplication with integer addition. It also reduces the number of iterations needed for an algorithm to converge. Finally, discretizing eliminates the need for precise random number generators.

The Pursuit Algorithm is among the quickest stochastic learning automata known to date. As such, it represents a natural candidate for discretization. Fortunately ε-optimality is preserved when the Pursuit Algorithm is discretized. In fact, in some difficult two-action environments the DPL_{RI} requires only about 50% of the number of iterations required for its continuous counter part. It required only 69% of the iterations required by the continuous Pursuit algorithm when learning in a ten-action environment. Indeed, the DPL_{RI} is probably the fastest absorbing learning automaton reported in the literature.

We are currently working on discretizing the ergodic pursuit algorithm, namely, Pursuit Algorithm 2 and are attempting to prove its asymptotically optimal properties. We believe that conclusions made in the comparison between the continuous and the discretized versions of the **absorbing** Pursuit Algorithm also apply to the (continuous and discretized) ergodic pursuit schemes.

ACKNOWLEDGEMENTS

We are very grateful to Mr. Valiveti who proofread the original manuscript for us and provided us with many helpful comments.

REFERENCES

[1] Baba, S., Soeda, S.T., and Sawaragi, Y., "An Application of Stochastic Automata to the Investiment Game," *Int. Journal Syst. Sci.*, vol. 11, no. 12, pp 1447-1457, Dec. 1980.

[2] Flerov, Y.A., "Some Classes of Multi-Input Automata", *Journal of Cybernetics*, Vol. 2, 1972, pp.112-122.

[3] Isaacson, D.L., and Madson, R.W., *Markov Chains : Theory and Applications*, Wiley, 1976.

[4] Karlin, S., Taylor, H. M., *A First Course on Stochastic Processes* , 2d ed., Academic Press, New York, 1974.

[5] Lakshmivarahan, S., *Learning Algorithms Theory and Applications*, Springer-Verlag, New York, 1981.

[6] Lakshmivarahan, S., *ε-Optimal Learning Algorithms - Non-Absorbing Barrier Type*, Technical Report EECS 7901, February 1979, School of Electrical Engineering and Computing Sciences, University of Oklahoma, Norman, Oklahoma.

[7] Lakshmivarahan, S., "Two Person Decentralized Team With Incomplete Information", *Applied Mathematics and Computation*, Vol. 8, pp.51-78, 1981.

[8] Lakshmivarahan, S., and Thathachar, M.A.L., "Absolutely Expedient Algorithms for Stochastic Automata", *IEEE Trans. on Syst. Man and Cybern.*, Vol. SMC-3, 1973, pp.281-286.

[9] Meybodi, M.R., *Learning Automata and Its Application to Priority Assignment in a Queueing System With Unknown Characteristics*, Ph.D. Thesis, School of Electrical Engineering and Computing Sciences, University of Oklahoma, Norman, Oklahoma.

[10] Narendra, K.S., and Thathachar, M.A.L., *Learning Automata*, Prentice-Hall, 1989.

[11] Narendra, K.S., and Thathachar, M.A.L., "Learning Automata -- A Survey", *IEEE Trans. on Syst. Man and Cybernetics*, Vol. SMC-4, 1974, pp.323-334.

[12] Narendra, K.S., and Thathachar, M.A.L., "On the Behaviour of a Learning Automaton in a Changing Environment With Routing Applications", *IEEE Trans. on Syst. Man and Cybern.*, Vol. SMC-10, 1980, pp.262-269.

[13] Narendra, K.S., Wright, E., and Mason, L.G., "Applications of Learning Automata to Telephone Traffic Routing", *IEEE Trans. on Syst. Man and Cybern.*, Vol. SMC-7, 1977, pp.785-792.

[14] Narendra, K.S., and Lakshmivarahan, S, "Learning Automata : A Critique", *Journal of Cybernetics and Information Sciences*, Vol. 1, 1987, pp.53-66.

[15] Oommen, B.J., and Hansen, E.R., "The Asymptotic Optimality of Discretized Linear Reward-Inaction Learning Automata", *IEEE Trans. on Syst. Man and Cybern.*, May/June 1984, pp.542-545.

[16] Oommen, B.J., and Christensen, J.P.R., "Epsilon-Optimal Discretized Linear Reward-Penalty Learning Automata", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-18, May/June 1988, pp. 451-458.

[17] Oommen, B.J., and Thathachar, M.A.L., "Multiaction Learning Automata Possessing Ergodicity of the Mean", *Information Sciences*, Vol. 35, No. 3, June 1985, pp. 183-198.

[18] Oommen, B. J., "Ergodic Learning Automata Capable of Incorporating *A priori* Information", *IEEE Trans. on Syst. Man and Cybern.*, Vol. SMC-17, July/August 1987, pp.717-723.

[19] Oommen, B.J., "Absorbing and Ergodic Discretized Two-Action Learning Automata", *IEEE Trans. on Syst. Man and Cybern.*, Vol. SMC-16, 1986, pp.282-296.

[20] Oommen, B.J., "A Learning Automaton Solution to the Stochastic Minimum Spanning Circle Problem", *IEEE Trans. on Syst. Man and Cybern.*, July/Aug. 1986, pp.598-603.

[21] Oommen, B. J., and Ma, D. C. Y., "Deterministic Learning Automata Solutions to the Equi-Partitioning Problem", *IEEE Transactions on Computers*, Vol. 37, January 1988, pp.2-14.

[22] Oommen, B.J., and Ma, D.C.Y., "Fast Object Partitioning Using Stochastic Learning Automata". Proceedings of the 1987 International Conference on Research and Development in Information Retrieval, New Orleans, June 1987.

[23] Ramesh, R., *Learning Automata in Pattern Classification*, M.E. Thesis, Indian Institute of Science, Bangalore, India, 1983.

[24] Sastry, P.S., *Systems of Learning Automata: Estimator Algorithms Applications*, Ph.D. Thesis, Dept of Electrical Engineering, Indian Institute of Science, Bangalore, India, June 1985.

[25] Thathachar, M.A.L., and Oommen, B.J., "Discretized Reward-Inaction Learning Automata", *Journal of Cybernetics and Information Sciences*, Spring 1979, pp.24-29.

[26] Thathachar, M.A.L., and Sastry, P.S., "A New Approach to Designing Reinforcement Schemes for Learning Automata", Proceedings of the IEEE Int. Conf. on Cybernetics and Systems, Bombay, India, January 1984.

[27] Thathachar, M.A.L., and Sastry, P.S., "A Class of Rapidly Converging Algorithms for Learning Automata", *IEEE Trans. on Syst. Man and Cybern.*, Vol SMC-15, pp. 168-175, January 1985.

[28] Thathachar, M.A.L., and Sastry, P.S., "Pursuit Algorithm for Learning Automata". Personal Communication to the first author of this paper available as an unpublished transcript. The principal ideas in the transcript appeared as part of the thesis in Ref.[24].

[29] Tsetlin, M.L., "On the Behaviour of Finite Automata in Random Media", *Automat. Telemekh.*(USSR), Vol.22, Oct. 1961, pp.1345-1354.

[30] Tsetlin, M.L., *Automaton Theory and the Modelling of Biological Systems*, New York and London, Academic, 1973.

[31] Tsypkin, Y.Z. and Poznyak, A.S., "Finite Learning Automata", *Engineering Cybernetics*, Vol.10, 1972, pp.478-490.

[32] Varshavskii, V.I., and Vorontsova, I.P., "On the Behaviour of Stochastic Automata With Variable Structure", *Automat. Telemek.*(USSR), Vol.24, 1963, pp.327-333.