A Hierarchical System of Learning Automata That Can Learn the Globally Optimal Path

M. A. L. THATHACHAR AND P. S. SASTRY

Department of Electrical Engineering, Indian Institute of Science, Bangalore 560012, India

ABSTRACT

Systems of learning automata have been studied by various researchers to evolve useful strategies for decision making under uncertainty. Considered in this paper are a class of hierarchical systems of learning automata where the system gets responses from its environment at each level of the hierarchy. A classification of such sequential learning tasks based on the complexity of the learning problem is presented. It is shown that none of the existing algorithms can perform in the most general type of hierarchical problem. An algorithm for learning the globally optimal path in this general setting is presented, and its convergence is established. This algorithm needs information transfer from the lower levels to the higher levels. Using the methodology of estimator algorithms, this model can be generalized to accommodate other kinds of hierarchical learning tasks.

1. INTRODUCTION

Making decisions without complete knowledge of the consequences of various choices open to the decision maker is an integral part of many intelligent problem solving systems. In a probabilistic setting, this problem of decision making under uncertainty has been studied by many researchers. To perform well in the face of such uncertainty, the system has to acquire some knowledge about the consequences of various choices. This acquisition of the relevant information can be posed as a learning problem wherein the system evaluates its experience and changes its behavior in the light of its experience in an endeavor to progressively perform better, tending towards the optimum value of some performance index. Such behavior was originally studied extensively in mathematical psychology [1,2]. In the systems-science literature, learning automata are among the models employed successfully to tackle the problem of decision making under uncertainty [3,4,5].

A learning automaton consists of a stochastic automaton interacting with a random environment. The automaton has a set of actions from which it is to choose one at each instant. To make this choice, the automaton maintains a probability distribution over the set of actions defined by its state at each
instant. For each action selected by the automaton, the environment responds with a random reward drawn from distributions unknown to the automaton. The automaton uses this response to update its state through a learning algorithm. There are many algorithms to make the automaton learn the optimal action through this interaction with the environment [4, 6].

Such automaton models have been used as adaptive decision making devices in many different applications [7–9]. While models consisting of a single automaton operating in a random environment are sufficient for simple problems like learning the scalar parameter of an unknown system, to tackle complex learning problems one is forced to deal with systems consisting of many automata [10, 11].

Hierarchical decomposition of problems is a commonly employed method in engineering sciences to tackle complexity. It is customary to divide a complex problem into subproblems and devise a scheme whereby a collection of subsystems, each working on a subproblem, can solve the whole problem. A number of complex decision problems have been tackled in such a hierarchical fashion [12]. Hence, one simple way of employing a number of automata to solve a problem collectively is to arrange them as a tree hierarchy. A tree structure allows for simple connections between the automata to make them work together.

In this paper we consider certain kinds of hierarchies of learning automata where the system gets responses from the environment at each level of the hierarchy. We develop a classification of these hierarchical systems based on the complexity of learning the optimal path. We shall show that the current learning algorithms for hierarchical systems are not capable of learning the globally optimal path in the most general class of hierarchies. Then we present an algorithm for learning this globally optimal path based on the methodology of estimator algorithms [11, 6].

Two-level hierarchical systems of learning automata were studied by Viswanathan and Narendra to cope with high-dimension parameter spaces in optimization problems and also for periodic nonstationary environments [13, 14]. Thathachar and Ramakrishnan have proposed a learning algorithm for an N-level hierarchy whereby the system is assured to be absolutely expedient [15]. Hierarchical systems with multiple reactions, the class of hierarchies considered in this paper, are introduced by them to model sequential learning processes where learning proceeds through a number of distinct stages [16]. There is considerable psychological evidence that in many complex learning tasks, the learning evolves in such a sequential fashion [17, 18]. It is shown in the next section that standard decision-theoretic problems like the Markovian decision processes [19] can be captured by these models.

We begin by introducing a hierarchical system of learning automata in Section 2. Section 3 discusses classification of hierarchies. In Section 4 we
present the learning algorithm and establish its convergence. Section 5 discusses some simulation results, and Section 6, the conclusions.

2. A HIERARCHICAL SYSTEM OF LEARNING AUTOMATA

We start by first defining a learning automaton. A learning automaton is a stochastic automaton in a feedback connection with a random environment. The following definitions are from [11] and are an extension of the standard ones [4].

A stochastic automaton can be defined as a quadruple \((A, Q, T, R)\) where \(A = \{\alpha_1, \ldots, \alpha_r\}\) is a finite set of actions. \(\alpha(k)\) denotes the action chosen by the automaton at \(k\). \(Q\) is the state of the automaton, with \(Q(k)\) denoting the state at \(k\). \(Q(k)\) includes a probability distribution \(p(k)\) over \(A\) defined by

\[
p_i(k) = \text{Prob}[\alpha(k) = \alpha_i].
\]

Define the \(r\)-dimensional simplex \(S\) by

\[
S = \left\{ p \in \mathbb{R}^r : p_i \geq 0, \sum_i p_i = 1 \right\}.
\]

Then \(p(k) \in S\) for all \(k\). For the models used in this paper \(Q(k) = (p(k), \hat{d}(k))\) when \(\hat{d}(k)\) is the estimate of \(d\), the vector of reward probabilities of the environment, at \(k\). \(R\) is the set of responses from the environment. \(\beta(k)\) denotes the response or reaction at \(k\), and \(\beta(k) \in R\) for all \(k\). The environment is said to be a \(P\)-, \(Q\)-, or \(S\)-model according as \(R = \{0, 1\}, R = \{\beta_1, \ldots, \beta_l\}\) \((l < \infty)\), or \(R = [0, 1]\). \(T\) is the learning algorithm or the reinforcement scheme defined by

\[
Q(k + 1) = T(Q(k), \alpha(k), \beta(k)).
\]

Thus \(T\) updates the state of the automaton in view of the action selected and reaction obtained.

The random environment is defined by the triple \((A, F, R)\), where \(A\) and \(R\) are as above and

\[
F = \{F_1, \ldots, F_r\}
\]

is a set of distribution functions such that

\[
\text{Prob}[\beta(k) \leq x | \alpha(k) = \alpha_i] = F_i(x).
\]
Define the vector of reward probabilities \( d = [d_1 \cdots d_r]' \) by

\[
d_i = E[\beta(k) | \alpha(k) = \alpha_i] = \int x dF_i(x).
\]

\( d_i \) is called the reward probability of the action \( \alpha_i \). If \( F_i \) and hence \( d_i \) are independent of \( k \), then the environment is said to be stationary. Otherwise it is nonstationary.

As mentioned in the previous section, the automaton has no knowledge of the functions \( F_i \). Through interaction with the environment it is to learn the action \( d_e \) (i.e., it is to evolve to a state with \( p_e = 1 \)) where the index \( e \) is such that \( d_e = \max_i \{d_i\} \). Then \( \alpha_e \) is called the optimal action.

A hierarchical system of automata is a tree structure where each node corresponds to an automaton and the arcs emanating from that node correspond to actions of that automaton. The root node corresponds to an automaton which will be referred to as the first-level or top-level automaton. Each of the actions of this automaton leads to a distinct automaton at the second level. In this way the structure can be extended to an arbitrary number of levels. A three-level hierarchy with two actions per automaton is shown in Figure 1.

To refer to various automata and their actions in the hierarchy, a general notation is introduced below. The following shorthand notation is used for

Fig. 1. A hierarchical system of learning automata.
making reference to general path in the tree:

\[ i_n = i_1 i_2 \ldots i_n, \]

\[ i_{n-1} j_n = i_1 i_2 \ldots i_{n-1} j_n, \]

\[ p_{i_n}(t) = p_{i_1 \ldots i_n}(t). \]

**NOTATION.** \( N \) denotes the number of levels in the hierarchy. \( A_0 \) denotes the first-level automaton. \( \beta_i \) denotes the reaction of the environment at the \( i \)th level, \( i = 1, \ldots, N. \)

**First level:**

\( A_0 \) is the automaton at the first level.

\{ \( \alpha_1, \ldots, \alpha_r \) \} is the set of all actions of \( A_0. \)

\{ \( d_1, \ldots, d_r \) \} is the set of reward probabilities for \( A_0. \)

\( p_1(k) = [p_1(k) \ldots p_r(k)]' \) is the action probability vector for \( A_0. \)

\( \hat{d}_1(k) = [\hat{d}_1(k) \ldots \hat{d}_r(k)]' \) are estimates of reward probabilities at \( k. \)

**\( n \)th level (2 \( \leq n \leq N):**

\( A_{i_{n-1}} = A_{i_1 i_2 \ldots i_{n-1}} \) is the automaton connected to the action \( \alpha_{i_1 i_2 \ldots i_{n-1}} \) of an automaton at the \( (n-1) \)th level, \( i_j \in \{1, \ldots, r\}, j = 1, \ldots, n-1. \)

\( \alpha_{i_n} \) is an action of \( A_{i_{n-1}}, i_n \in \{1, \ldots, r\}. \)

\( d_{i_n} \) is the reward probability of \( \alpha_{i_n}. \)

\( p_{i_n}(k) \) is the action probability for \( \alpha_{i_n}. \)

\( \hat{d}_{i_n}(k) \) is the estimate of \( d_{i_n} \) at \( k. \)

\( p_{i_n}(k) \) is the action probability vector of \( A_{i_{n-1}} \) at \( k. \)

Now referring to Figure 1, one can easily follow this notation. \( A_0 \) is the first-level automaton, whose actions are \( \alpha_1 \) and \( \alpha_2. \) The second-level automaton connected to, say, action \( \alpha_2 \) is \( A_2, \) whose actions are \( \alpha_{21} \) and \( \alpha_{22}. \) The third-level automaton connected to, say, \( \alpha_{21} \) is \( A_{21}, \) whose actions are \( \alpha_{211} \) and \( \alpha_{212}. \) Thus the above notation allows for easy reference to various automata in the hierarchy.

In a hierarchy, each action at the last level has a unique path connecting it to the automaton at the top level (i.e., to \( A_0 \).) The product of the probabilities of the actions lying on this path is called the path probability and is denoted by

\[ \pi_{i_{n-1}}(k) = \pi_{i_1 i_2 \ldots i_n}(k) \]

\[ = p_{i_1}(k) p_{i_2}(k) \ldots p_{i_n}(k), \]
and

\[ \sum_{i_1, \ldots, i_N} \pi_{i_N}(k) = 1 \quad \text{for all } k. \]

Let the indices \( m_1, m_2, \ldots, m_N \) be defined by

\[ d_{m_1} d_{m_2} \cdots d_{m_N} = \max_{i_1, \ldots, i_N} \{ d_{i_1} d_{i_2} \cdots d_{i_N} \}. \quad (1) \]

Then the sequence of actions \( \alpha_{m_1}, \alpha_{m_2} \cdots \alpha_{m_N} \) is said to constitute the optimal path. The product of the reward probabilities of the actions on the optimal path is the maximum over all paths in the hierarchy. Now, \( \pi_{i_N} \) is the probability of the optimal path. As explained in Section 6, the algorithm to be presented can easily be modified to learn the optimal path under other kinds of definitions for optimal path as well.

The operation of this hierarchical learning system is as follows. At any instant \( k \), the first-level automaton chooses an action, say \( \alpha_{i_1} \), at random according to its action probability distribution. This activates the automaton \( A_{i_1} \) at the second level, which chooses an action, say \( \alpha_{i_1i_2} = \alpha_{i_2} \). This in turn activates the automaton \( A_{i_2} \), and so on. When the whole path is traversed, the environment responds with a set of \( N \) reactions, one for each level. Using these reactions, the system updates the states of its automata through the learning algorithm and the cycle repeats. The problem is to design learning algorithms so that this process converges to a state where the path probability of the optimal path is close to unity.

Now we show that a general problem in decision making can be well captured by the model presented above.

Consider a simple Markovian decision process in which there are finitely many stages [19]. Let \( N \) be the number of stages. There is a subject who is to make a choice of one of finitely many actions at each stage. For each action that he takes, he gets a reward from the environment. The environment would be modeled by a Markov process. Assume that the process always starts in the same state, say \( i_0 \), of the environment. Each choice of action by the decision maker determines a transition matrix for the environment. So, after the decision is made, the environment changes state according to the chosen transition matrix. For every action, there is a reward matrix that determines the reward to the subject based on the state transition of the environment. This process goes on for all \( N \) stages and then stops. The objective is to obtain a sequence of actions under which the total gain to the subject through the rewards received is maximum. Many sequential decision processes can be put in this framework (see [19] for more details).
Now, the corresponding learning problem for such sequential decision process can be posed as follows. Assume that the transition matrices of the environment for various actions and the associated reward structures are completely unknown. Also assume that the subject gets no information about the state of the process at any of the intermediary stages except at the first and the rewards are random. Under these circumstances, the system is to learn an optimal sequence of actions for the $N$ stages through repeated interaction (e.g. through simulation or prototype models) with the environment.

The $N$-stage decision problem can be modelled by a $N$-level hierarchy.

Let $P(j_n)$ represent the transition matrix of the environment under the action $j_n$ (which, by the notation given earlier, would be an action at the $n$th stage) with elements $P_{ij_n}(j_n)$. Let $R(j_n)$ with elements $r_{ij_n}(j_n)$ be the corresponding reward matrix. The hierarchical model would function as follows. The automaton $A_0$ will make the decision, say $\alpha_i$, at the first stage. Then the automaton $A_{i_1}$ at second level of the hierarchy will be responsible for the decision at the second stage, and so on. The reward due to the state transition under $\alpha_{i_1}$ would be the reaction to $A_0$, and similarly for the others. To keep the notation simple, consider a three-stage problem with a binary reward structure, i.e., a $P$-model environment. Then it is easy to see that the reward probabilities at different levels are given by the following expressions ($M$ is the number of states for the environment):

$$d_{j_1} = \text{Prob}[A_0 \text{ gets reward} | A_0 \text{ chose } \alpha_{i_1}]$$

$$= \sum_{i_1=1}^{M} P_{ij_1}(j_1) r_{ij_1}(j_1), \quad j_1 = 1, \ldots, r,$$

$$d_{j_1j_2} = \text{Prob}[A_{i_1} \text{ gets reward} |$$

$$A_{j_1} \text{ is activated and it selected } \alpha_{i_1j_2}],$$

$$= \sum_{i_1=1}^{M} \sum_{i_2=1}^{M} P_{ij_1}(j_1) P_{i_1j_2}(j_2) r_{ij_2}(j_2),$$

$$d_{j_1j_2j_3} = \sum_{i_1=1}^{M} \sum_{i_2=1}^{M} \sum_{i_3=1}^{M} P_{ij_1}(j_1) P_{i_1j_2}(j_2) P_{i_2j_3}(j_3) r_{ij_3}(j_3). \quad (2)$$

It is easy to see that in the hierarchical setup for this problem described above, these will be the unknown reward probabilities for the automata $A_0$, 

Now, define a path reward probability,
\[ \chi_{j_1 j_2 j_3} = d_{j_1} d_{j_2} d_{j_3}. \]

Since the rewards are random, the objective is to maximize the probability of getting a reward at each of the \(N\) stages. Hence the optimal path can be defined to be consisting of actions \(\alpha_{m_1}, \alpha_{m_1 m_2}, \alpha_{m_1 m_2 m_3}\) such that
\[ \chi_{m_1 m_2 m_3} = \max_{j_1, j_2, j_3} \{ \chi_{j_1 j_2 j_3} \}. \]

Now it is easy to see that the hierarchical model described earlier can be used to solve the learning problem.

3. TYPES OF HIERARCHIES

In this section hierarchical systems of learning automata are classified into three types based on the nature of the optimal path. This classification is motivated by the nature of the conditions needed for ensuring convergence of different learning algorithms.

Recall from Section 2 that the sequence of actions \(\alpha_{m_1}, \alpha_{m_2}, \ldots, \alpha_{m_N}\) constituting the optimal path is such that
\[ d_{m_1} d_{m_2} \cdots d_{m_N} = \max_{j_1, \ldots, j_N} \{ d_{j_1} d_{j_2} \cdots d_{j_N} \}. \]

The action \(\alpha_{i_n}\) is called the optimal action of the automaton \(A_{i_{n-1}}\) if
\[ d_{i_n} = \max_j \{ d_{i_{n-1} j} \}. \]

The action \(\alpha_{i_n}\) is said to be optimal at \(n\)th level if
\[ d_{i_n} = \max_{j_1, \ldots, j_n} \{ d_{j_n} \}. \]

The classification of the hierarchies can now be explained as follows.

Hierarchy of Type I. A hierarchical system of learning automata is said to be of Type I if the actions constituting the optimal path are also individually optimal at their respective levels.

Consider the hierarchical system shown in Figure 2. The reward probability of each action is shown on the arc representing the action. The optimal path is
A HIERARCHICAL SYSTEM OF LEARNING AUTOMATA

Fig. 2. A type-I hierarchical system.

Hierarchical System of Learning Automata

A hierarchical system of learning automata is said to be of type II if the actions constituting the optimal path are also the optimal actions of their respective automata.

Consider the hierarchical system shown in Figure 3. The optimal path is given by \( \alpha_1\alpha_{11} \). It is easy to see that \( \alpha_1 \) is the optimal action of \( A_0 \), and \( \alpha_{11} \) is the optimal action of \( A_1 \). Hence this is a type-II system. But this hierarchy is not of type I, because \( \alpha_{11} \) is not optimal at the second level.

From the definitions given above, it is clear that every type-I hierarchy is also a type-II hierarchy. Because of the example given in Figure 3, this inclusion is proper. That is, the class of type-II hierarchies is strictly a superset of the class of type-I hierarchies.

Fig. 3. A type-II hierarchy that is not type I.
Hierarchy of type III. Any general hierarchy is said to be of type III. Consider the three-level hierarchy shown in Figure 4. The optimal path is $a_1a_{11}$. This is not a type-II hierarchy, because the optimal action of $A_1$ is $a_{12}$ and not $a_{11}$.

Thus, we have the following inclusion relationships among different types. Every type-I hierarchy is a type-II hierarchy, and every type-II hierarchy is a type-III hierarchy. Further, each of these inclusions is proper, as is evident from the examples given in Figures 3 and 4. It may be noticed that the situation here is similar to the classification of phrase-structured grammars.

The absolutely expedient algorithm proposed in [16] converges in the sense of $\epsilon$-optimality to the optimal path if the hierarchy is of type I. There the updating has to proceed sequentially from top to bottom because the updating at one level depends on the updated probability in the level above. In [6] the authors have introduced an estimator algorithm for a hierarchical system of learning automata which can learn the optimal path in any type-II hierarchy. In this algorithm, the updatings at different levels are decoupled and can proceed concurrently. But this algorithm cannot learn the optimal path in a general type-III hierarchy that is not type II. Consider the Markovian decision problem discussed earlier. From Equation (2), it is obvious that the matrices $P(j)$ and $R(j)$ have to satisfy some extra conditions for the problem to fit into the framework of a type-II hierarchy. In general, it would be type III. Hence, the algorithm in [6] cannot be assured of success in such a problem.
In the next section we present an algorithm that can learn the optimal path in a general type-III hierarchical system. The restriction on the optimal path in a type-II hierarchy can be viewed as a local optimality condition. In a type-II hierarchy, every action on the optimal path is locally optimal in the sense that it is the optimal action of its automaton. If one goes down the hierarchy starting from $A_0$ and picks the optimal actions of the automata encountered along the way, the path picked in Figure 4 is $a_1a_{12}a_{121}$, which is not the optimal path. But a similar procedure would yield the optimal path in a type-II hierarchy (this, of course, is not a solution technique for the learning problem, because the reward probabilities are unknown). In view of this, the optimal path in a type-III hierarchy will be referred to as the globally optimal path.

The algorithm presented in the next section can learn the globally optimal path in a hierarchy. But unlike the algorithm in [6], this needs information transfer from the lower levels of the hierarchy to the higher levels. Since any hierarchical decomposition of a problem assumes more abstraction at higher levels, this is a natural flow direction for information.

4. THE LEARNING ALGORITHM

In this section we present a learning algorithm to be used by a hierarchical system of learning automata for learning the globally optimal path in a hierarchy. As stated earlier, this algorithm needs some information transfer between the levels of the hierarchy. The following definitions are needed to explain the nature of this information transfer.

**Definition 1.** For each action $a_{i_n}$ at the $n$th level, its cumulative reward probability $E_{i_n}$ is defined recursively as follows:

$$E_{i_n} = d_{i_n},$$

$$E_{i_n} = d_{i_n} \max \left\{ E_{i_{n+1}} \right\}, \quad 1 \leq n \leq N - 1. \quad (3)$$

It is easy to see that $E_{i_n}$ can be expressed as

$$E_{i_n} = d_{i_n} \max \left\{ d_{i_{N-n}} \right\}. \quad (4)$$

Thus the cumulative reward probability $E_{i_n}$ of $a_{i_n}$ is the product of the reward probabilities on a path which starts with $a_{i_n}$ and is optimal in the part of the hierarchy starting with $a_{i_n}$.
Now we define the concept of path optimal action for each automaton in the hierarchy. In type-II hierarchies we have seen that the optimal path is composed of actions that are optimal actions of their respective automata. In type-III hierarchies the analogous concept is the path optimal action of an automaton.

**Definition 2.** For each automaton $A_{n-1}$ at the $n$th level, the action $a_{n-1,m'}_t$ is called its *path optimal action*, where the index $m'_n$ is defined by

$$E_{i_n-1,m'_n} = \max_{j_n} \{ E_{i_n-1,j_n} \}.$$  

Using (3) and (5), one can write $E_{i_n}$ as

$$E_{i_n} = d_{i_n} E_{i_n,m'_n+1}.$$  

Now let the sequence of actions $\alpha_{m_1}, \alpha_{m_2}, \ldots, \alpha_{m_N}$ constitute the optimal path defined by (1). Then defining the path reward probability as in Section 2, we obtain

$$x_{m_N} = \max_{i_1, i_2, \ldots, i_N} \{ d_{i_1} d_{i_2} \cdots d_{i_N} \}.$$  

Hence, if $\alpha_{m_1}$ is the first action on the optimal path as defined by (1), then the index $m_1'$ which defines the path optimal action of the top-level automaton is identical to $m_1$. This property can be seen to extend to all other automata on the optimal path as well. (This point is further clarified while proving Theorem 1). Thus, the optimal path defined by the maximum of the product of its constituent reward probabilities is always such that the actions constituting the optimal path are path optimal actions of their respective automata.

Hence, to learn the optimal path each automaton should converge to its path optimal action. This is achieved by the learning algorithm presented below.
A HIERARCHICAL SYSTEM OF LEARNING AUTOMATA

ALGORITHM I. Let $\alpha_{i_{1}}, \alpha_{i_{2}}, \ldots, \alpha_{i_{N}}$ be the actions chosen at instant $k$. Let $\beta_{1}, \beta_{2}, \ldots, \beta_{N}$ be the reactions at the various levels. Then the steps in the updating at the $n$th level ($1 \leq n \leq N$) are as given below:

1. $p_{i_{n-1}j_{n}}(k + 1) = p_{i_{n-1}j_{n}}(k) - \lambda_n \left[ f(\hat{E}_{i_{n}}(k)) - f(\hat{E}_{i_{n-1}j_{n}}(k)) \right]$

$$\times \left[ s_k(i_{n}, i_{n-1}j_{n}) p_{i_{n-1}j_{n}}(k) \right.$$

$$+ \frac{s_k(i_{n-1}j_{n}, i_{n})(1 - p_{i_{n-1}j_{n}}) p_{i_{n}}(k)}{r - 1} \right], \quad j_{n} \neq i_{n}, \quad (8)$$

$$p_{i_{n}}(k + 1) = 1 - \sum_{j_{n} \neq i_{n}} p_{i_{n-1}j_{n}}(k + 1).$$

$O < \lambda_n < 1$ are constants, $1 \leq i \leq N$,

$f: [0,1] \rightarrow [0,1]$ is a monotonically increasing function, and

$$s_k(q_{n}, t_{n}) = \begin{cases} 1 & \text{if } \hat{E}_{q_{n}}(k) > \hat{E}_{t_{n}}(k), \\ 0 & \text{otherwise}. \end{cases} \quad (9)$$

2. $Z_{i_{n}}(k + 1) = Z_{i_{n}}(k) + 1,$

$$Z_{i_{n-1}j_{n}}(k + 1) = Z_{i_{n-1}j_{n}}(k), \quad j_{n} \neq i_{n}, \quad (10)$$

$$R_{i_{n}}(k + 1) = R_{i_{n}}(k) + \beta_{n},$$

$$R_{i_{n-1}j_{n}}(k + 1) = R_{i_{n-1}j_{n}}(k), \quad j_{n} \neq i_{n},$$

and

$$\hat{d}_{i_{n-1}j_{n}}(k + 1) = \frac{R_{i_{n-1}j_{n}}(k + 1)}{Z_{i_{n-1}j_{n}}(k + 1)}, \quad 1 \leq j_{n} \leq r. \quad (11)$$
3. 
\[ \hat{E}_{i_{n-1}, j_n}(k+1) = F_{i_{n-1}, j_n}(k) \hat{d}_{i_{n-1}, j_n}(k+1), \quad 1 \leq n \leq N-1, \]
\[ \hat{E}_{i_{N-1}, j_n}(k+1) = \hat{d}_{i_{N-1}, j_n}(k+1), \]
\[ F_{i_{n-1}}(k+1) = \max_{j_n} \left\{ \hat{E}_{i_{n-1}, j_n}(k+1) \right\}, \quad 1 \leq n \leq N. \]

4. 
\[ p_{j_{n-1}}(k+1) = p_{j_{n-1}}(k), \quad \hat{E}_{j_{n-1}}(k+1) = \hat{E}_{j_{n-1}}(k), \]
and
\[ F_{j_{n-1}}(k+1) = F_{j_{n-1}}(k) \]
for all other automata (i.e. \( j_{n-1} \neq i_{n-1} \)) at the \( n \)th level. Here \( \hat{E}_{j_{n-1}} = [\hat{E}_{i_{n-1}} \cdots \hat{E}_{i_{n-1}r}]' \).

**Remark 1.** The updating for each automaton consists essentially of three steps. The first step is to get \( p_{j_{n-1}, i_{n-1}}(k+1) \), using the equations (8). This updating depends on the sign of \( \hat{E}_{i_{n}} - \hat{E}_{i_{n-1}, j_{n}} \), and the functions \( s_k(\cdot, \cdot) \) defined by (9) are used to make the choice depending on the sign. \( \hat{E}_{i_{n}} \) is the estimated cumulative reward probability. The second step in the algorithm is to obtain \( \hat{d}_{i_{n-1}, j_{n}} \) using (11). This is obtained simply as a sample mean, and the vectors \( R \) and \( Z \) are used only for notational convenience. At each instant only one element of the vector \( \hat{d} \) is changed. The final step is to obtain \( \hat{E}_{i_{n-1}, j_{n}} \) using Equation (12). Since \( E_{i_{n-1}, j_{n}} \) depends on reward probabilities in other parts of the hierarchy besides those of \( A_{i_{n-1}} \), the recursive definition of \( E_{i_{n}} \) given by (3) is used to obtain the estimate \( \hat{E}_{i_{n-1}, j_{n}} \). For this an information transfer between the levels is needed, and the quantity feedback from level \( n \) to the level above is termed \( F_{i_{n-1}} \). It is easy to see that only one component of the vector \( \hat{E} \) is changed at each instant.

**Remark 2.** Comparing Algorithm I with the one presented in [6], the essential difference is that \( \hat{E}_{i_{n}} \) is used in place of \( \hat{d}_{i_{n}} \). Using the algorithm in [6], the hierarchy converges to the optimal path in type-II hierarchies where each automaton on the optimal path is to converge to the action with highest \( d_{i_{n}} \). To learn the globally optimal path, each automaton on the optimal path has to converge to an action with highest \( E_{i_{n}} \). Hence, given the methodology of
estimator algorithms [11], it is natural to use, in place of \( d_i^n \), an estimate \( \hat{E}_i^n \) of \( E_i^n \). The information transfer in the form of \( F_i^n \) is needed for this estimation. Since \( \hat{E}_i^n(k + 1) \) is made dependent on \( F_i^n(k) \), all the updatings can still proceed concurrently. But after the updating cycle is over, there has to be an information transfer between adjacent levels. The nature of this transfer is quite interesting. Each automation at the \((n + 1)\)st level, say \( A_{(n+1)} \), abstracts the information regarding the reward structure of the hierarchy below it into a single quantity \( F_i^n \) and passes it on to the automaton directly above it, that is, \( A_{(n+1)} \). Thus, this setup captures the natural idea behind hierarchical decomposition of any problem into subproblems.

The convergence result for this algorithm is stated in the following theorem. The algorithm converges under the following assumption:

A1. There is a unique optimal path in the hierarchy.

**Theorem 1.** Consider a hierarchical system of learning automata using Algorithm I and functioning in an environment satisfying A1. Then, given any \( \epsilon > 0 \), \( \delta > 0 \), there exist \( \lambda_1^*, \ldots, \lambda_N^* \), all greater than zero, and \( N_0 < \infty \) such that for all \( \lambda_i \in (0, \lambda_i^*) \), \( 1 \leq i \leq N \),

\[
\text{Prob} \left[ \prod_{j=1}^{N} p_{m_j}(k) - 1 < \epsilon \right] > 1 - \delta \quad \text{for all} \quad k \geq N_0. \tag{13}
\]

To prove this theorem we need two general results on estimator algorithms, both of which are stated below. The proofs of these propositions can be found in [6,11].

**Proposition 1.** Consider an estimator algorithm with updating given by (8). Suppose there exist an index \( q_n \) and a time instant \( K_0 \) such that

\[
\hat{E}_{i_{n-1}q_n}(k) > \hat{E}_{i_n}(k), \quad i_n \neq q_n, \quad k \geq K_0.
\]

Then

\[
\lim_{k \to \infty} p_{i_{n-1}q_n}(k) = 1 \quad \text{w.p. 1.}
\]

**Proposition 2.** Consider an estimator algorithm with updating given by (8). Given any \( M < \infty \), \( \delta > 0 \), there exist \( \lambda^* > 0 \) and \( K_0 < \infty \) such that for all
\[ \lambda_n \in (0, \lambda^*) \],

\[ \text{Prob} \left[ \text{All actions of } A_{i_{n-1}} \text{ are chosen} \right. \]

\[ \text{at least } M \text{ times each before the automaton has completed} \]

\[ k \text{ iterations} \]

\[ > 1 - \delta \quad \forall k \geq K_0. \]

**Proof of Theorem 1.** Let \( \chi_{i_N} \) denote the path reward probability as in Section 2. That is,

\[ \chi_{i_N} = d_{i_1}d_{i_2} \cdots d_{i_N}. \]

The reward probability of the optimal path is

\[ \chi_{m_N} = d_{m_1}d_{m_2} \cdots d_{m_N}. \]

By A1,

\[ \chi_{m_N} > \chi_{i_N} \quad \forall i_N \text{ s.t. } i_j \neq m_j \text{ for at least one } j. \]

Let \( \Delta > 0 \) be the difference between the highest two path reward probabilities. By (4) and (7),

\[ E_{m_1} = \chi_{m_N}, \]

where \( E_{m_1} \) is the cumulative reward probability of action \( \alpha_{m_1} \) of the top level automaton \( A_0 \). Also the difference between the highest two cumulative reward probabilities of actions of \( A_0 \) will be at least \( \Delta \). By (12), the estimated cumulative reward probabilities of actions of \( A_0 \) are given by

\[ \hat{E}_{i_t} = \hat{d}_{i_{t_1}} \max \left\{ \hat{d}_{i_{t_2}}, \ldots, \hat{d}_{i_{t_N}} \right\}. \]  \( (14) \)

Hence there exists a \( \Delta_0 > 0 \) (which will be of the order of \( \Delta/N \)) such that if the estimated reward probability of any action, \( \hat{d}_{i_N} \), in the hierarchy is locked within a \( \Delta_0 \)-neighborhood of its true value, then \( \hat{E}_{i_t} \) will be in a \( \Delta/2 \)-neighbor-
hood of its true value. This implies, by the definition of $\Delta$, that $\hat{E}_{m_1} > \hat{E}_{i_1}$ for all $i_1 \neq m_1$. Thus we have

$$\text{Prob}\left[ \hat{E}_{m_1} > \hat{E}_{i_1}, i_1 \neq m_1 \mid |\hat{d}_{i_k} - d_{i_k}| < \Delta_0 \forall i_n, 1 \leq n \leq N \right] = 1. \quad (15)$$

We first show that if $\hat{E}_{m_1} > \hat{E}_{i_1} \forall i_1 \neq m_1$, then each automaton on the optimal path converges to its path optimal action. Then we show that there exists a choice of $\lambda_i$'s such that all $\hat{d}_{i_k}$ can be locked in a $\Delta_0$-neighborhood around their true values with arbitrarily large probability in a finite time. Finally we show that each automaton on the optimal path converging to its path optimal action is equivalent to (13).

Define a set of events $B_n(k)$

$$B_n(k) = \left[ \hat{E}_{m_{n+1}}(t) > \hat{E}_{i_{n+1}}(t), i_{n+1} \neq m_{n+1}, t \geq k \right], \quad 0 \leq n \leq N - 1.$$  

Thus $B_0(k)$ is the event of $\hat{E}_{m_1} > \hat{E}_{i_1}$ for all time after $k$. By Proposition 1, given $\epsilon > 0, \delta > 0$, there exists a $K_1 < \infty$ such that for all $n$, $0 \leq n \leq N - 1$,

$$\text{Prob}\left[ |p_{m_{n+1}}(k) - 1| < \epsilon \mid B_n(K_0) \right] > 1 - \delta \quad \forall k \geq K_1 \geq K_0. \quad (16)$$

Consider $A_{m_1}$, the second automaton on the optimal path. Its path optimal action is $a_{m_1m_2}$. The estimated cumulative reward probability of an action of $A_{m_1}$ is given by

$$\hat{E}_{m_1} = \hat{d}_{m_1j} \max \left\{ \hat{d}_{m_1j_1} \cdots \hat{d}_{m_1j_{n-2}} \right\}. \quad (17)$$

When the estimated cumulative reward probabilities of actions of $A_0$ are within $\Delta/2$ of their true values, we have

$$\hat{d}_{m_1} \cdots \hat{d}_{m_N} > \hat{d}_1 \cdots \hat{d}_N \quad \forall i_N \text{ s.t. } i_j \neq m_j \text{ for at least one } j. \quad (18)$$

Taking $i_1 = m_1$ in the above,

$$\hat{d}_{m_2} \cdots \hat{d}_{m_n} > \hat{d}_{m_1j} \cdots \hat{d}_{m_1j_{n-1}} \quad \forall j_{n-1} \text{ s.t. } j_k \neq m_k \text{ for at least one } k. \quad (19)$$
Taking \( j_1 = m_2 \) in the above, we obtain

\[
\hat{d}_{m_2} \cdots \hat{d}_{m_N} = \max_{i_1, \ldots, i_{N-2}} \left\{ \hat{d}_{i_1 \cdots i_{N-2}} \right\}.
\]  

(20)

(17)–(20) together imply that

\[
\hat{E}_{m_1 m_2} > \hat{E}_{m_1 j}, \quad j \neq m_2,
\]

whenever \( \hat{E}_{m_1} > \hat{E}_i \forall i \neq m_1. \)

Thus the event \( B_1 \) is implied by the event \( B_6 \). Similarly we can show, considering each automaton on the optimal path in turn, that each of \( B_j \) is implied by \( B_i \). Hence (16) can be rewritten as

\[
\text{Prob} \left[ \left| p_{m_{n+1}} - 1 \right| < \epsilon \left| B_0 \left( K_0 \right) \right| \right] > 1 - \delta \quad \forall k \geq K_1 > K_0, \quad 0 \leq n \leq N - 1.
\]  

(21)

This completes the first part of the proof.

Consider any estimated reward probability \( \hat{d}_{i_2} \) in the hierarchy. Since it is estimated as the sample mean, \( \hat{d}_{i_2} \rightarrow d_{i_2} \) if \( \alpha_{i_2} \) is chosen infinitely often. Since \( \Delta_0 > 0 \) is a constant, given any \( \delta > 0 \), there exists \( K_0 \) such that

\[
\text{Prob} \left[ \left| \hat{d}_{i_2}(k) - d_{i_2} \right| < \Delta_0 \right] > 1 - \delta \quad \forall k \text { s.t. } \eta_{i_2}(k) > K_0,
\]  

(22)

where \( \eta_{i_2}(k) \) is the number of times the action \( \alpha_{i_2} \) is tried till \( k \). Note that \( K_0 \) is a constant independent of \( i_2 \), because the \( \Delta_0 \) required is the same for all actions and there are only finitely many actions. Consider the automata at \( N \)th level. Each of \( \hat{d}_{i_2} \) will converge in the sense of (22) if each action is chosen at least \( K_0 \) times. By Proposition 2, there exist \( \lambda_N^* > 0 \) and \( M_N < \infty \) such that with \( \delta > 0 \) given, \( \lambda_N \in (0, \lambda_N^*) \),

\[
\text{Prob} \left[ \text{all actions of } A_{i_{N-1}} \text{ are chosen} \right] > 1 - \delta
\]  

(23)

for all \( k \) such that \( Z_{A_{i_{N-1}}}(k) \geq M_N \). Here \( Z_{A_{i_{N-1}}}(k) \) gives the number of times the automaton \( A_{i_{N-1}} \) is chosen till \( k \). This last condition is due to the fact that not every automaton in the hierarchy is activated at every instant.
Now, (22) and (23) imply that
\[
\text{Prob}\left[|\hat{d}_{i_N}(k) - d_{i_N}| < \Delta_0 \right] > 1 - \delta
\]
for all \( k \) such that \( ZA_{i_{N-1}}(k) \geq M_N \) and \( \lambda_N \in (0, \lambda^*_N) \). Now consider the automaton at the \((N - 1)\)th level. Each of \( \hat{d}_{i_{N-1}} \) will converge in the sense of (22) if each action at this level is tried at least \( K_0 \) times. Since each automaton at the \( N \)th level is connected to an action at the \((N - 1)\)th level and since \( M_N > K_0 \), if each action at the \((N - 1)\)th level is tried \( M_N \) times, then \( \hat{d}_{i_{N-1}} \) will converge and \( ZA_{i_{N-1}}(k) \geq M_N \) will be satisfied as well. Thus, once again applying Proposition 2, we can obtain the following: Given \( \delta > 0 \), there exist \( \lambda^*_N > 0, \lambda^*_{N-1} > 0, \) and \( M_{N-1} < \infty \) such that
\[
\text{Prob}\left[|\hat{d}_{i_N}(k) - d_{i_N}| < \Delta_0 \right] > 1 - \delta
\]
and
\[
\text{Prob}\left[|\hat{d}_{i_{N-1}}(k) - d_{i_{N-1}}| < \Delta_0 \right] > 1 - \delta
\]
for all \( k \) such that \( ZA_{i_{N-1}}(k) \geq M_{N-1} \) and \( \lambda_N \in (0, \lambda^*_N), \lambda_{N-1} \in (0, \lambda^*_{N-1}) \). Proceeding inductively, we can obtain the following: Given \( \delta > 0 \), there exist \( \lambda^*_i > 0 \) and \( M_0 < \infty \) such that for all \( \lambda_i \in (0, \lambda^*_i), 1 \leq i \leq N, \)
\[
\text{Prob}\left[|\hat{d}_{i}(k) - d_{i}| < \Delta_0 \right] > 1 - \delta \quad \forall k \geq M_0, \quad 1 \leq n \leq N, \quad i_j \in \{1, \ldots, r\},
\]
(24)
since \( ZA_0(k) = k \).

We can now combine this with (15), (21) as follows: Given \( \epsilon > 0 \), there exist \( \lambda^*_i > 0, 1 \leq i \leq N, M_0 < \infty \) such that
\[
\text{Prob}\left[|p_{m_{n+1}}(k) - 1| < \epsilon \right]
\]
\[
\geq \text{Prob}\left[|p_{m_{n+1}}(k) - 1| < \epsilon | B_0(K_0) \right]
\times \text{Prob}\left[B_0(K_0) | |\hat{d}_{i_n}(k) - d_{i_n}| < \Delta_0 \forall i_n, 1 \leq n \leq N, k \geq K_0 \right]
\times \text{Prob}\left[|\hat{d}_{i_n}(k) - d_{i_n}| < \Delta_0 \forall i_n, 1 \leq n \leq N, k \geq K_0 \right]
\geq (1 - \delta)^2
\]
(25)
for all \( k \geq M_0, \lambda_i \in (0, \lambda^*_i) \), by using (15), (21), and (24).
Thus we have established the existence of \( \lambda^* \) such that each automaton on the optimal path converges to its path optimal action. We now show that (25) is equivalent to (13).

Consider the automata \( A_p \) and \( A_{m_1} \), whose path optimal actions are \( \alpha_{m_1} \) and \( \alpha_{m_2} \) respectively. (25) implies that there exists \( M < \infty \) such that

\[
\Pr\left[p_{m_1}(k) > 1 - \epsilon \right] > 1 - \frac{\delta}{2}
\]

and

\[
\Pr\left[p_{m_2}(k) > 1 - \epsilon \right] > 1 - \frac{\delta}{2} \quad \text{for all } k \geq M.
\]

(26)

For some \( t > M \), define events

\[
A_{1t} = \left\{ p_{m_1}(t) < 1 - \epsilon \right\},
\]

\[
A_{2t} = \left\{ p_{m_2}(t) < 1 - \epsilon \right\}.
\]

By (26)

\[
\Pr(A_{1t}) < \frac{\delta}{2} \quad \text{and} \quad \Pr(A_{2t}) < \frac{\delta}{2}.
\]

Let \( A_t \) be the set of sample paths given by

\[
A_t = \left\{ p_{m_1}(t) p_{m_2}(t) < 1 - \epsilon_1 \right\},
\]

where \( 1 - \epsilon_1 = (1 - \epsilon)^2 \). Then

\[
A_t \subseteq (A_{1t} \cup A_{2t}).
\]

Now, we have

\[
\Pr\left[p_{m_1}(t) p_{m_2}(t) \geq 1 - \epsilon_1 \right] = 1 - \Pr(A_t)
\]

\[
\geq 1 - \Pr(A_{1t} \cup A_{2t})
\]

\[
\geq 1 - \Pr(A_{1t}) - \Pr(A_{2t})
\]

\[
\geq 1 - \delta.
\]
Since this is true for any \( t > M \), we have

\[
\text{Prob} \left[ \left| P_{m_1}(t) P_{m_2}(t) - 1 \right| < \epsilon \right] > 1 - \delta \text{ for all } t \geq M.
\]

Proceeding inductively, one can now shown (13). This completes the proof of Theorem 1.

5. SIMULATION RESULTS

In this section we describe a computer simulation of Algorithm I on an example. A three-level hierarchy with each automaton having five actions is chosen for the simulation. A \( P \)-model environment is used, with reward probabilities of actions on the optimal path all set to 0.8 and those of the remaining actions chosen at random from a uniform distribution over [0, 0.75]. The optimal path is chosen to be \( a_1 a_{11} a_{11} \). Since Algorithm I can function in a general hierarchy, this hierarchy is now converted to a type-III hierarchy that is not type II. For this, one of the reward probabilities of \( A_1 \), namely that of \( a_{12} \), is changed to 0.9. All the reward probabilities of \( A_{12} \) are adjusted to retain \( a_1 a_{11} a_{111} \) as the optimal path. Now one of the actions on the optimal path, namely \( a_{12} \), is not the optimal action of its automaton, and hence this is not a type-II hierarchy.

The results obtained using Algorithm I are shown in Tables 1 and 2. Table 1 shows the number of iterations needed for the path probability of the optimal path to be greater than 0.99, averaged over 25 experiments. Table 2 shows a typical run. In the implementation of the algorithm, we took \( f(x) = x \).

From the results obtained, it is seen that the rate of convergence is good. The absolutely expedient algorithm of Thathachar and Ramakrishnan [16] takes over 10,000 iterations to converge on a similar three-level hierarchy. As mentioned earlier, this algorithm is effective only in type-I hierarchies and needs sequential updatings. The algorithm presented in [6], applicable only for type-II hierarchies, takes over 2000 iterations to converge on a similar three-level problem.

<table>
<thead>
<tr>
<th>TABLE 1</th>
<th>Performance of Algorithm I</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda_1 )</td>
<td>0.05</td>
</tr>
<tr>
<td>( \lambda_2 )</td>
<td>0.1</td>
</tr>
<tr>
<td>( \lambda_3 )</td>
<td>0.15</td>
</tr>
<tr>
<td>Average number of iterations for convergence:</td>
<td>1715</td>
</tr>
</tbody>
</table>
TABLE 2

<table>
<thead>
<tr>
<th>Iteration number</th>
<th>Path probability of optimal path</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.008</td>
</tr>
<tr>
<td>1000</td>
<td>0.6015</td>
</tr>
<tr>
<td>2000</td>
<td>0.9973</td>
</tr>
<tr>
<td>3000</td>
<td>0.9999</td>
</tr>
</tbody>
</table>

6. CONCLUSIONS

In this paper we have considered hierarchical systems of learning automata where the system gets responses from the environment at each level of the hierarchy. We have classified such hierarchies into three types based on the restrictions imposed on the optimal path. The classification is one of progressive generalization. We have presented a learning algorithm that can learn the globally optimal path in the most general type of hierarchical system.

It is interesting to compare Algorithm I presented in this paper with the algorithm for the hierarchy given in [6]. In [6] each automaton on the optimal path converges to its optimal action, and in a type-II hierarchy this results in the convergence of the system to the optimal path. To remove this local-optimality restriction so that learning can result even in a type-III environment, it is realized that one needs information regarding the reward structure at the lower levels of the hierarchy. This information is characterized in the form of cumulative reward probabilities, and it is shown that each automaton has to converge to the action with highest cumulative reward probability for the hierarchy to converge to the globally optimal path. Hence we have used estimates of cumulative reward probabilities instead of estimates of the reward probabilities as in [6]. These estimates are obtained through a simple information-transfer structure using a recursive formulation of the cumulative reward probabilities.

Thus Algorithm I exemplifies the flexibility in the methodology of estimator algorithms discussed in [6,11]. This is due to the structure of the updating equations, which allows for feeding back relevant information from the environment.

This methodology can also handle other types of generalizations, for example, in the definition of the optimal path. In this paper the optimal path was defined through the maximum of the product of reward probabilities. The objective is to maximize \[ \text{Prob}[\beta_1 = 1 \text{ and } \cdots \text{ and } \beta_N = 1] \] over all paths.
Suppose in some hierarchical learning problem the objective is to maximize the probability of $f(\beta_1, \ldots, \beta_N)$ being high for a given $f$. By properly formulating cumulative reward probabilities one can extend Algorithm I to handle this case. In Equation (14), as long as $E_{i, n}$ is a continuous function of $d_{i, n}$, the argument about the existence of $\Delta_0$ is valid and the rest of the proof would simply follow. A concrete example would be to define the optimal path through the maximum sum of reward probabilities, which would be useful in finding minimum-cost paths in random networks.

Another kind of generalization that can be easily accommodated is the multiteacher environments introduced by Baba [20]. In [6] we have shown how an estimator algorithm for a single automaton can be very easily modified to handle the problem of multiteacher environments. The idea is to formulate a proper combination of all the responses from the environment into an estimate that replaces the estimate of the reward probability (see [6] for details). By an identical change in the updating equations at each level, Algorithm I can learn in a multiteacher environment as well.

REFERENCES


Received 26 December 1985; revised 1 February 1987