Hierarchical Discretized Pursuit Nonlinear Learning Automata with Rapid Convergence and High Accuracy

Georgios I. Papadimitriou

Abstract—**In this paper, a new absorbing multiaction learning automaton that is epsilon-optimal is introduced. It is a hierarchical discretized pursuit nonlinear learning automaton that uses a new algorithm for positioning the actions on the leaves of the hierarchical tree. The proposed automaton achieves the highest performance (speed of convergence, central processing unit (CPU) time, and accuracy) among all the absorbing learning automata reported in the literature up to now. Extensive simulation results indicate the superiority of the proposed scheme. Furthermore, it is proved that the proposed automaton is epsilon-optimal in every stationary stochastic environment.**

Index Terms—**Hierarchical learning automaton, pursuit learning algorithm, nonlinear output function, epsilon-optimal learning automation, positioning algorithm**

I. Introduction

Adaptive learning is one of the main fields of artificial intelligence. Learning automaton is one of the most powerful tools in this scientific area. It is a finite state machine that interacts with a stochastic environment trying to learn the optimal action of this environment via the following learning process (Fig. 1). The automaton chooses one of the actions according to a probability vector, which at every instant contains the probability of choosing each action. The chosen action triggers the environment that responds with an answer (reward or penalty) according to the reward probability of the chosen action. The automaton takes into account this answer and modifies its state by means of a transition function. The new state of the automaton corresponds to a new probability vector given by a function, called output function. A learning automaton is one that learns the action that has the maximum probability to be rewarded and that ultimately chooses this action more frequently than other actions.

Manuscript received March 23, 1991; revised March 22, 1993.

The author is with the Department of Computer Engineering, University of Patras, 26500 Patras, Greece, and the Computer Technology Institute, 26110 Patras, Greece.

IEEE Log Number 9212683.

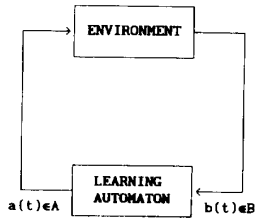

Fig. 1. A learning automaton that interacts with a stochastic environment.

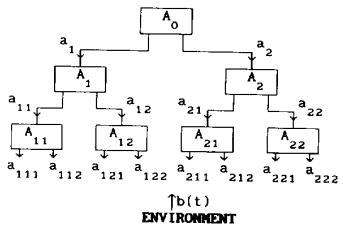

Fig. 2. A hierarchical learning automaton interacting with a stochastic environment.

Learning automata that keep estimates of the actions' reward probabilities and at every time instant increase the choice probability of the action that has the highest estimated reward probability are called pursuit learning automata [7].

With respect to their Markovian representation, learning automata are classified into two main categories: ergodic [2], [3] or automata-possessing absorbing barriers [2], [3], [5], [7]. The ergodic automata converge with a distribution independent of the initial state. On the other hand, the automata with absorbing states, after a number of finite steps, get locked (converge) into a particular action. If the reward probabilities of the actions are stable (stationary environment), absorbing automata are preferred.

Another way to classify learning automata is according to the values that the action probabilities can take. By this view, a learning automaton can be characterized as a continuous or a discretized one. In the former case, the action probabilities can take any value in the [0, 1] interval. In the latter, the action probabilities can take values from a finite set only. In other words, the [0, 1] interval is divided into a finite number of subintervals. If these subintervals are all of equal length, the automaton is characterized as a linear one; if not, it is a nonlinear one.

An efficient way to combine several learning automata in order to construct another one that has a much larger number of actions is to join them in a hierarchical mode [4], [5]. The automata that compose the hierarchy are called subautomata and are connected in a tree structure (Fig. 2). Every action of each subautomaton (except of the last level ones) corresponds to a subautomaton of the next level of the tree. The hierarchy operates as follows. The subautomaton on the top of hierarchy selects an action according to its probability vector. The selected action corresponds to a subautomaton of the next level of hierarchy. This subautomaton activates and selects an action that, in turn, corresponds to a subautomaton of the next level. This operation is repeated until the last level of hierarchy is reached. The action selected by the subautomaton of the last level interacts directly with the environment. The reaction of the environment is used to update the action probabilities at all the levels of hierarchy.

In this paper, we present a new absorbing hierarchical discretized pursuit nonlinear (HDPN) learning automaton that is epsilon-optimal in every stationary stochastic environment. Simulation results indicate that for a given accuracy, the proposed HDPN learning automaton is the fastest converging absorbing automaton reported in the literature.

The structure of this paper is as follows. The proposed HDPN learning automaton is presented in Section II. The formal definition of the HDPN in Section III is followed by the proof of its epsilon-optimality in Section IV. Extensive simulation results that indicate the superiority of HDPN's performance are presented in Section V. Finally, a brief discussion of the proposed scheme closes the paper in Section VI.

II. The HDPN Learning Automaton

The main features of the proposed HDPN learning automaton are the learning algorithm, the nonlinear output function, the optimal selection of the resolution parameters of various level subautomata, and the new positioning algorithm that positions the actions on the leaves of the hierarchical tree in an optimal way. These features are extensively analyzed and discussed in the rest of this section.

A. The Learning Algorithm

The proposed learning algorithm operates as follows. Initially the estimator vector is initialized by executing a Z iterations per action trial period (where Z is usually a small integer). After the completion of the trial period, the automaton sets its subautomata to their middle state (probability= 0.5). Then the hierarchical automaton selects a path that leads from the top level to the bottom one, according to the probability vectors of the subautomata. The last level's action interacts with the environment.

If the feedback from the environment is a penalty, then the probability vector does not change, and the operation described above is repeated. This will slow the convergence when the action chosen is not the optimal one.

If the feedback is a reward, then the automaton increases by one step the probabilities of the path that leads from the top subautomaton to the leaf of the hierarchical tree that corresponds to the action (let a_b) that has the highest estimated reward probability ($d'_b(t) = \max_i\{d'_i(t)\}$).

The operation described above is repeated until a path from the top hierarchy level to the last one obtains a probability equal to unity. In this situation, every subautomaton of the path has converged. The formal definition of the learning algorithm described above is presented in Section III.

B. The Output Function

The following is the output function of the two-action discretized subautomata [1] that compose the HDPN:

$$G(k) = \begin{cases} A(k/n)^e & 0 \le k \le n/2, \\ 1 - A((n-k)/n)^e & n/2 < k \le n, \end{cases}$$

where $e = 0.01, A = 2^{0.01}/2, k$ is the state of the automaton, and n is an internal parameter of the automaton, which is called "resolution parameter."

According to the above output function, the subintervals of the probability space [0, 1] are small near the center (0.5) and get bigger as we move from the center to the extreme states. This division improves the accuracy of the two-action scheme. Therefore, the accuracy of the hierarchical scheme is also improved.

C. The Resolution Parameters of Various Level Subautomata

A subautomaton of level i has the half probability to be updated than a subautomaton of the previous level $(i-1)$. If $n(i)$ denotes the resolution parameter of the ith level subautomata, then by selecting $n(i) = n(i-1)/2$, we give to all subautomata of the several levels of hierarchy the same chances to converge. In this way, we maximize the probability of concurrent convergence, and consequently we

TABLE I
THE NEW POSITIONING ALGORITHM VS. RANDOM POSITIONING IN A 16-ACTION ENVIRONMENT

	HDPN USING THE NEW POSITIONING ALGORITHM		HDPN WITH RANDOM POSITIONING OF THE ACTIONS	
n	ACCURACY	ITERAT.	ACCURACY	ITERAT.
1024	0.999	1298	0.995	1282
512	0.994	765	0.992	757
256	0.973	487	0.963	477
128	0.941	333	0.908	326
64	0.873	250	0.857	248
32	0.801	206	0.768	204

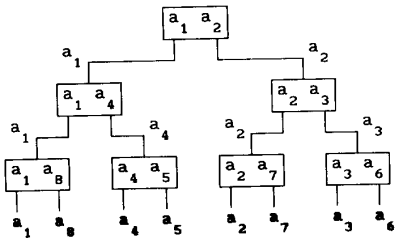

Fig. 3. An example of the new positioning algorithm.

maximize the parallelism of the scheme, which leads to an optimal performance.

D. The Positioning of the Actions on the Leaves of the Hierarchical Tree

The positioning of the actions on the leaves of the hierarchical tree is a very significant part of the automaton's operation, because it influences the accuracy of the hierarchical scheme. The positioning algorithm that we propose uses the initial information of the estimator (after the trial period) in order to position the actions on the leaves of the hierarchical tree as efficiently as possible.

The purpose of the proposed positioning algorithm is the execution of comparisons between actions that both have relatively high estimated reward probability, which (comparisons) are more "critical" for the accuracy of the scheme, at the higher level's subautomata, because the latter are more powerful (have a larger resolution parameter) than the subautomata of the lower levels.

As simulation results show (Table I), the use of the proposed positioning algorithm increases the accuracy of the scheme significantly, and its speed of convergence remains invariant. The following is the new positioning algorithm.

Step 1: Sort the actions according to their initial estimates of reward probabilities.

Step 2: Put actions i and $K - i + 1$ together (to be compared by a two-action subautomaton) at the current level of hierarchy, starting from the last level, where K is the number of actions of the current level of hierarchy.

Step 3: For each pair of compared actions, pass to the previous level of hierarchy that action that has a higher initial estimate of reward probability, and repeat steps 2 and 3 until the top level is reached.

An example of the above positioning algorithm is presented in Fig. 3. Assume that actions are sorted according to the initial estimates of their reward probabilities. Thus, action a_1 has the highest initial estimate of reward probability, a_2 has the second one, and so forth. The comparison between actions a_1 and a_2, a comparison that is probably critical for the accuracy of the scheme, is performed by the top level subautomaton that is the most powerful one. Less critical comparisons (such as the one between actions a_4 and a_5) or less difficult comparisons (as the one between actions a_1 and a_8) are performed by the lower level's subautomata.

The proposed positioning algorithm is based on the initial estimates of reward probabilities. Therefore, its efficiency is proportional to the accuracy of the initial estimates. Even when the accuracy of the initial estimates is low, however, the proposed positioning algorithm is more efficient than the random positioning of the actions on the leaves of the hierarchical tree.

III. FORMAL DEFINITION OF THE HDPN

Before presenting the formal definition of the HDPN learning automaton, we have to define our notation.

1) Let N be the number of levels of hierarchy. Thus, for an r-action HDPN learning automaton, we have $N = \lceil \log_2 r \rceil$.
2) n is the resolution parameter of the top level subautomaton.
3) To simplify long subscripts, we use the following notation [4]. Denote $i_1 i_2 \cdots i_n$ by $\mathbf{i}_n$. Also, denote $m_1 m_2 \cdots m_n$ by $\mathbf{m}_n$. Thus, $\mathbf{i}_1 = i_1, \mathbf{i}_2 = i_1 i_2, \mathbf{m}_1 = m_1, \mathbf{m}_2 = m_1 m_2$, and so on.

The HDPN learning automaton is defined as a quintuple $\langle A, B, Q, T, G \rangle$, where $A = \{a_{i_1 i_2 \cdots i_N} : i_x = 1, 2 \text{ for } x = 1, .., N\}$ is the set of actions, implying their position on the leaves of the hierarchical tree.

$B = \{0, 1\}$ is the set of the environment's responses (where 1 symbolizes a REWARD, and 0 symbolizes a PENALTY response). $b(t) \in B$ denotes the response at time instant t.

$Q(t) = \{S(t), D'(t)\}$ is the state of the automaton where $S(t) = \{S_0(t)\} \cup \{S_{i_1 i_2 \cdots i_k}(t) : k = 1, 2, \cdots, N-1$ and $i_x = 1, 2$ for $x = 1, \cdots, k\}$ is the set of the states of the subautomata that compose the hierarchical scheme (where the state of a two action subautomaton is defined similarly to the state of a two-action discretized automaton [2] and $S_{i_1 i_2 \cdots i_k}(t) \in \{0, 1, \cdots, n/(2^{k-1})\}$).

Furthermore, $D'(t) = \{d'_{i_1}(t_{i_2 \cdots i_N}) : i_x = 1, 2 \text{ for } x = 1, \cdots, N\}$ is the estimator vector that contains the estimated reward probabilities of the actions, as shown below:

$$d'_{i_N}(t) = \frac{\text{times } \textbf{REWARDED} \text{ action } a_{i_N} \text{ up to time instant } t}{\text{times } \textbf{SELECTED} \text{ action } a_{i_N} \text{ up to time instant } t}. \quad (1)$$

G: is the output function discussed in Section II-B. If the resolution parameter of the top (1st level) subautomaton is n, then an Lth-level subautomaton has a resolution parameter equal to $n/2^{L-1}$. Thus, for an Lth-level subautomaton that is at state S, the output function is as follows:

$$G(L, S) = \begin{cases} A\left(\frac{S \, 2^{L-1}}{n}\right)^{\epsilon}, & 0 \le S \le n/2^L, \\ 1 - A\left(\frac{(n/2^{L-1}) - S}{n/2^{L-1}}\right)^{\epsilon}, & n/2^L < S \le n/2^{L-1}. \end{cases}$$

We define the set of subautomata that compose the hierarchical scheme as $E = \{A_0\} U \{A_{i_1 i_2 \cdots i_L} : L = 1, \cdots, N-1$ and $i_x = 1, 2$ for $x = 1, \cdots, L\}$. A subautomaton $A_{\mathbf{i}_{k-1}} \in E (1 \le k \le N)$ selects action $a_{\mathbf{i}_k}$ at time instant t with the following probability:

$$P_{\mathbf{i}_k}(t) = G(k, S_{\mathbf{i}_{k-1}}(t))(-1)^{i_k - 1} + (i_k - 1). \quad (2)$$

Thus, the HDPN selects action a_{i_N} at time t with the following probability:

$$\mathbf{Pr}\{a(t) = a_{i_N}\} = \prod_{k=1}^{N} P_{\mathbf{i}_k}(t). \quad (3)$$

T: is the learning algorithm discussed in Section VI-A. Its algorithmic description is presented below (assuming that $m_0 = 0$).

1) According to the action probabilities of the subautomata that compose the HDPN learning automaton, select a path from the

TABLE II
HDPN VS. DPA IN A 32-ACTION ENVIRONMENT

	n/n*	ACCURACY	ITERATIONS	UPDATINGS
HDPN	1024	0.994	1556	3093
	512	0.973	998	1720
	256	0.942	692	953
	128	0.893	523	526
DPA	400	0.986	2910	56165
	200	0.971	2068	38024
	50	0.917	1038	15371
	25	0.888	753	9129

TABLE III
HDPN VS. DPA IN A 64-ACTION ENVIRONMENT

	n/n*	ACCURACY	ITERATIONS	UPDATINGS
HDPN	2048	0.990	3173	7551
	1024	0.972	2034	4188
	512	0.940	1425	2388
	256	0.907	1081	1363
DPA	400	0.951	6629	268293
	200	0.939	4637	178211
	100	0.922	3199	112653
	50	0.904	2279	71090

top-level subautomaton to a leaf of the hierarchical tree. By using that action (say, $a(t) = a_{i_N}$), which corresponds to this leaf, trigger the environment.

2) Receive the feedback $b(t) \in \{0, 1\}$ from the environment.
3) Compute the new estimated reward probability $d'_{i_N}(t)$ of action a_{i_N} as it is defined by relation (1).
4) If $b(t) = 0$ (* Penalty *), then GOTO Step 1.
5) Select action a_{m_N} that has the highest estimate $d'_{m_N}(t)$ of reward probability among all other actions.
6) Update the states $S_{m_0}(t), S_{m_1}(t), \cdots, S_{m_{N-1}}(t)$ of all subautomata that compose the path from the top-level subautomaton to the leaf that corresponds to action a_{m_N}, by using the following rule (for $k = 0, 1, \cdots, N-1$):

if $P_{m_{k+1}}(t) < 1$ then $S_{m_k}(t+1) := S_{m_k}(t) + (-1)^{(m_{k+1}+1)}$.

7) Compute the new action probabilities $P_{m_1}(t+1), \cdots, P_{m_N}(t+1)$ of the subautomata that compose the path that leads to action a_{m_N} by using relation (2).
8) If the following is true:

$$\prod_{k=1}^{N} P_{m_k}(t+1) = 1,$$

THEN CONVERGE to action a_{m_N} ELSE GOTO Step 1.

IV. Proof of Epsilon-Optimality

Theorem: In every stationary stochastic environment, the HDPN learning automaton is ε-optimal. In other words, if action a_b is the optimal one, then, given $\varepsilon > 0$ and $\delta > 0$, there exists a $n_0 < \infty$ and a $t_0 < \infty$ such that for all $t > t_0$ and $n > n_0$, $\mathbf{Pr}\{|p_b(t) - 1| < \varepsilon\} > 1 - \delta$.

Proof: The proof is given in the Appendix (Theorem 3).

V. Simulation Results

Simulation was performed to compare the speed of convergence and accuracy of HDPN and DPA [7]. We decided to compare the proposed scheme with DPA, because it is the fastest converging absorbing automaton reported in the literature up to now.

For both schemes, the automaton was said to have converged whenever the probability of choosing an action was exactly unity. The two automata that are under comparison were simulated operating in environments of 32 and 64 actions. For all simulations, the reward probability of the optimal action was 0.77, and the reward probabilities of all other actions were equally spaced in the interval [0.6, 0.15]. Thus, in the r-action case, we have $d_1 = 0.77$ and $d_i = 0.6 - (i-2)\delta$ for $i = 2, 3, \cdots, r$ where $\delta = (0.6 - 0.15)/(r-2)$.

Before starting the algorithm, initial estimates for the estimator vector D' were obtained by selecting each action 10 times, as was done by Thathachar and Sastry [8] and Oommen and Lanctot [7]. These extra iterations were then included in the total number of iterations until the algorithm converged. The ensemble average results are shown in Tables II and III.

The value of the resolution parameter appears at the first column of each table. In the HDPN case, the resolution parameter n is defined as the resolution parameter of the top-level subautomaton. In the DPA case, the resolution parameter n^* is defined by the relation $\Delta = \frac{1}{rn^*}$, where r is the number of actions and Δ is step size [7] of the probability updating.

The accuracy corresponding to the resolution parameter of the first column appears at the second column of each table. Note that the accuracy is $E[p_b(\infty)]$, where $p_b(\infty)$ is the probability of selecting the optimal action a_b, whereas time tends to infinity. The mean number of iterations required for convergence is appeared at the third column of each table.

Finally, the mean CPU time required for convergence, measured in the number of probability updatings (as was done by Thathachar and Ramakrishnan [4]) appears at the fourth column of each table. From these simulation results, it becomes clear that in any case, the HDPN performs much better than the DPA scheme. For example, operating in the 64-action environment and for the same accuracy, HDPN requires, on average, only 30% of the iterations and only 3% of the CPU time that DPA requires for convergence.

VI. Conclusion

The slow convergence rate of learning automata has been a limiting factor in their applications. This paper presents a hierarchical learning automaton that achieves the highest speed of convergence and the highest accuracy among all absorbing learning automata reported in the literature up to now. Furthermore, it was proved that the proposed learning automaton is epsilon-optimal in every stationary stochastic environment.

The proposed automaton uses a new algorithm for positioning the actions on the leaves of the hierarchical tree in an optimal way. The use of this algorithm leads to a significant improvement of the accuracy of the hierarchical scheme. Since the new positioning algorithm is applicable to any hierarchical scheme, it can be the base of a new generation of accurate and rapid hierarchical learning automata.

Appendix

Lemma 1: The modification of the choice probability of an action of a two-action subautomaton with the output function $G(k) = A(k/m)^e$ (where m is the resolution parameter, $A = 2^e/2$ and $e \leq 1$) when its state changes by one step is as follows:

$$|\Delta P| = \alpha \frac{1}{m^e} \text{ where } 0 < \alpha \leq 1.$$

Proof: $|\Delta P| = P(k+1) - P(k) = A\left(\frac{k+1}{m}\right)^e - A\left(\frac{k}{m}\right)^e = \frac{A}{m^e}\left((k+1)^e - k^e\right) = \alpha \frac{1}{m^e}$ where $\alpha = A((k+1)^e - k^e)$. Because $e \leq 1$, we have $0 < (k+1)^e - k^e \leq 1 \Rightarrow 0 < \alpha \leq A$. Furthermore, $e \leq 1 \Rightarrow A = 2^e/2 \leq 1$. Thus, $0 < \alpha \leq A \Rightarrow 0 < \alpha \leq 1$, so lemma 1 has been proved.

Lemma 2: The change of the choice probability of an action $a_j (j = 1, \cdots, r)$ after an iteration of the HDPN learning automaton is as follows:

$$\Delta p_j = \sum_{k=1}^{N} \frac{1}{n^{\epsilon k}} \sum_{i=1}^{\binom{N}{k}} b_{ik} P_i^k,$$
$$-2^{\frac{N(N+1)\epsilon}{2}} \le b_{ik} \le 2^{\frac{N(N+1)\epsilon}{2}} \text{ and } 0 \le P_i^k \le 1,$$

where P_i^k is the ith of the $\binom{N}{k}$ products of the $N-k$ probabilities of a total of N probabilities that compose the path of the hierarchical tree that leads to action a_j. The probabilities of the actions that compose this path are numbered from the top to the bottom as $P_1, P_2, \cdots, P_N$. E.g., for a tree of height 3 ($N = 3$), we have $P_1^1 = P_1 P_2 P_3, P_1^2 = P_1 P_2, P_2^2 = P_1 P_3$, etc.

Proof: The resolution parameter of an Lth level subautomaton is $\frac{n}{2^{L-1}}$. From lemma 1 is derived that for any action a_j, we have the following equation:

$$p_j(t+1) = \prod_{L=1}^{N} \left(P_L + \frac{\alpha_L \, 2^{(L-1)\epsilon}}{n^\epsilon} \right) = p_j(t) + \sum_{k=1}^{N} \frac{1}{n^{\epsilon k}} \sum_{i=1}^{\binom{N}{k}} b_{ik} P_i^k.$$

Thus, $\Delta p_j = \sum_{k=1}^{N} \frac{1}{n^{\epsilon k}} \sum_{i=1}^{\binom{N}{k}} b_{ik} P_i^k$, where $-1 \le \alpha_L \le 1$ and b_{ik}, P_i^k are as defined above.

Lemma 3: The learning algorithm of HDPN guarantees the following result:

$$p_i(t) \ge p_i(0) - t \sum_{k=1}^{N} \left(\frac{1}{n^\epsilon} \right)^k \binom{N}{k} 2^{\frac{N(N+1)\epsilon}{2}}.$$

Proof: The lemma is derived directly from lemma 2, satisfying $0 \le P_i^k \le 1$ and $|b_{ik}| \le 2^{\frac{N(N+1)\epsilon}{2}}$.

Theorem 1: For any given constants $\delta > 0$ and $M < \infty$, there exists $n_0 < \infty$ and $t_0 < \infty$ such that under the HDPN learning algorithm for all resolution parameters $n > n_0$ and all time instants $t > t_0$: $\mathbf{Pr}\{$each action has been chosen at least M times at time $t\} \ge 1 - \delta$.

Proof: Let us define the random variable R_i^t as the number of times the action a_i was chosen up to time t. Then we must prove that $\mathbf{Pr}\{R_i^t > M\} \ge 1 - \delta$. This relation is equivalent to the following:

$$\mathbf{Pr}\{R_i^t \le M\} \le \delta. \tag{4}$$

Since the events $\{R_i^t = k\}, \{R_i^t = j\}$ are mutually exclusive for $j \ne k$, relation (4) is equivalent to the following one:

$$\sum_{k=1}^{M} \mathbf{Pr}\{R_i^t = k\} \le \delta. \tag{5}$$

For any time instant t, $p_i(t) \le 1$. So, by Lemma 3, it is derived that $\mathbf{Pr}\{a_i$ is not chosen at time $t\} = 1 - p_i(t) \le 1 - p_i(0) + t \sum_{k=1}^{N} \left(\frac{1}{n^\epsilon} \right)^k \binom{N}{k} 2^{\frac{N(N+1)\epsilon}{2}}$. So, the probability that action a_i is chosen at most M times among t choices has the upper bound given by the following relation:

$$\mathbf{Pr}\{R_i^t \le M\} \le \sum_{j=1}^{M} \binom{t}{j} (1)^j \left[1 - p_i(0) + t \sum_{k=1}^{N} \left(\frac{1}{n^\epsilon} \right)^k \binom{N}{k} 2^{\frac{N(N+1)\epsilon}{2}} \right]^{t-j}.$$

Consider an arbitrary term of the above sum, say, $j = m$. It suffices to prove that the m^{th} term of this sum is less than δ/M, or, equivalently, that M times the mth term is less than δ. Thus, we have to prove the following:

$$M \binom{t}{m} (1)^m \left[1 - p_i(0) + t \sum_{k=1}^{N} \left(\frac{1}{n^\epsilon} \right)^k \binom{N}{k} 2^{\frac{N(N+1)\epsilon}{2}} \right]^{t-m} \le \delta.$$

It is known that $\binom{t}{m} \le t^m$. Using this inequality, we have to prove the following:

$$M t^m \left[1 - p_i(0) + t \sum_{k=1}^{N} \left(\frac{1}{n^\epsilon} \right)^k \binom{N}{k} 2^{\frac{N(N+1)\epsilon}{2}} \right]^{t-m} \le \delta. \tag{6}$$

Now, in a way analogous to the one used in reference [7], we have the following. In order to get the L.H.S. of this term to be less than δ as t increases, the result of the following equation:

$$Q = \left[1 - p_i(0) + t \sum_{k=1}^{N} \left(\frac{1}{n^\epsilon} \right)^k \binom{N}{k} 2^{\frac{N(N+1)\epsilon}{2}} \right], \tag{7}$$

must be strictly less than unity. We have the following condition:

$$Q \le \left[1 - p_i(0) + t \left(\frac{1}{n^\epsilon} \right) 2^{\frac{N(N+1)\epsilon}{2}} \sum_{k=1}^{N} \binom{N}{K} \right] = \left[1 - p_i(0) + t \left(\frac{1}{n^\epsilon} \right) 2^{\frac{N(N+1)\epsilon}{2}} (2^N - 1) \right]. \tag{8}$$

If we select $n \ge \left[\frac{2t}{p_i(0)} 2^{\frac{N(N+1)\epsilon}{2}} (2^N - 1) \right]^{1/\epsilon}$, then (8) gives $Q < 1$. For example, if we select the following:

$$n = \left[\frac{2t}{p_i(0)} 2^{\frac{N(N+1)\epsilon}{2}} (2^N - 1) \right]^{1/\epsilon}, \tag{9}$$

then (8) becomes as follows:

$$Q \le \left[1 - p_i(0) + \frac{p_i^{(0)}}{2} \right] = 1 - \frac{p_i^{(0)}}{2} < 1.$$

Thus, $Q < 1$. With this value of n, (6) simplifies to $\mathbf{Pr}\{R_i^t \le M\} \le M t^m Q^{t-m}$, where Q is as defined above, and, consequently, $0 < Q < 1$. It remains to show the following:

$$\lim_{t \to \infty} (M t^m Q^{t-m}) = 0. \tag{10}$$

With n and Q as defined above (relations (9) and (7), correspondingly), and, using l'Hopital's rule, we have the following equation:

$$\lim_{t \to \infty} (M t^m Q^{t-m}) = M \lim_{t \to \infty} \frac{t^m}{(1/Q)^{t-m}} = M \lim_{t \to \infty} \frac{m!}{(\ln(1/Q))^m (1/Q)^{t-m}} = 0.$$

Thus, the limit of relation (10) is equal to 0 as t tends towards infinity, whenever relation (9) is satisfied. Thus, since the limit exists, for every action a_i, there is a time instant $t(i)$ such that for all $t > t(i)$, (6) is less than δ when (9) is satisfied. Now set the following condition:

$$n(i) = \left[\frac{2t(i)}{p_i(0)} 2^{\frac{N(N+1)\epsilon}{2}} (2^N - 1) \right]^{1/\epsilon}.$$

It remains to be shown that (6) is satisfied for all $n > n(i)$ and for all $t > t(i)$. This is trivial because as n increases, the L.H.S. of (6) is monotonically decreasing, and so the inequality (6) is preserved. Also,

for any $n > n(i)$ and $t > t(i)$, because $R_i^{t(i)} \geq M \Rightarrow R_i^t \geq M$, by the laws of probability, $\mathbf{Pr}\{R_i^t \geq M\} \geq \mathbf{Pr}\{R_i^{t(i)} \geq M\}$.

Thus, in this case, too, the inequality (6) is preserved. Thus, for any action a_i, we get the following equation:

$$\mathbf{Pr}\{R_i^t \leq M\} \leq \delta \quad \text{whenever } t > t(i) \text{ and } n > n(i).$$

We can repeat this argument for all the actions. So, we can define t_0 and n_0 as follows:

$$t_0 = \max_{1 \leq i \leq r} \{t_i\} \text{ and } n_0 = \max_{1 \leq i \leq r} \{n_i\}.$$

Thus, for all i, it is true that for all $t > t_0$ and $n > n_0$, the quantity $\mathbf{Pr}\{R_i^t \leq M\} \leq \delta$, and the theorem has been proved.

Theorem 2: If there exists an index b and a time instant $t_0 < \infty$ such that $d_b'(t) > d_j'(t)$ for all j, such that $j \neq b$ and all $t > t_0$, then there exists an integer n_0 such that for all resolution parameters $n > n_0$, $p_b(t)$ tends to 1 with probability 1, as t tends toward ∞.

Proof: Assume the N two-action subautomata that compose the path from the root of the hierarchical tree to the leaf that corresponds to the "optimal" action a_b ("optimal path"). It suffices to show that each one of these subautomata "converges" to its "optimal" action (the action that leads to the "optimal path") with probability 1. If this condition is satisfied, then from relation (3) is derived that the probability of selecting action a_b is equal to 1, so that the HDPN has converged to the "optimal" action a_b.

Let $\mathbf{A}$ be an $\mathbf{L}$th-level subautomaton, and let a_1^*, a_2^* its actions. Now let us define the probabilities with which subautomaton A selects actions a_1^* and a_2^* at time t as $P_1^*(t)$ and $P_2^*(t)$, correspondingly. Further, let us define the state of the $\mathbf{A}$ subautomaton at time instant t as $S^*(t)$. Thus, $P_1^*(t) = G(L, S^*(t))$ and $P_2^*(t) = 1 - G(L, S^*(t))$. With no loss of generality, assume that action a_1^* is the one that leads to the "optimal path." It remains to prove that $P_1^*(t)$ tends to unity with probability 1 as t tends toward infinity.

Since $\mathbf{A}$ is an Lth-level subautomaton, its "resolution parameter" is $n(L) = n/2^{L-1}$.

It is given that $d_b'(t) > d_j'(t)$ for all j, such that $j \neq b$ and all $t \geq t_0$. Therefore, for all $t > t_0$, we get the following equation:

$$S^*(t+1) = \min\{S^*(t)+1, n(L)\} \quad \text{if } b(t) = 1(\mathbf{REWARD})$$
$$S^*(t+1) = S^*(t) \quad \text{if } b(t) = 0(\mathbf{PENALTY})$$

If $S^*(t) = n(L)$, then $P_1^*(t) = 1$, and the result is trivially proved.

Assume that the $\mathbf{A}$ subautomaton has not converged to action a_1^*. If $b(t) = 1$, we can generally write $S^*(t+1) = S^*(t) + z_t$, where z_t is equal to 1 when $S^*(t) < n(L)$ and equal to 0 when $S^*(t) = n(L)$. Let us define d_t^{av} as the probability of receiving a reward at time t. Thus, $d_t^{av} = \sum_{i=1}^r d_i p_i(t)$. Obviously, $d_t^{av} > 0$. We have the following equation:

$$E[S^*(t+1)|Q(t), S^*(t) \neq n(L)] = d_t^{av}(S^*(t) + z_t) + (1 - d_t^{av})S^*(t) = S^*(t) + d_t^{av} z_t.$$

Since the previous terms have an upper bound of $n(L)$, $E[S^*(t+1)|Q(t), S^*(t) \neq n(L)]$ is bounded. Hence, $\sup_{t \geq 0} E[S^*(t+1) \mid Q(t), S^*(t) \neq n(L)] < \infty$. Thus, $E[S^*(t+1) - S^*(t) \mid Q(t)] = d_t^{av} z_t > 0$ for all $t \geq t_0$, implying that $S^*(t)$ is a submartingale. By the submartingale convergence theorem [9], $\{S^*(t)\}_{t \geq t_0}$ converges, and so as $t \to \infty$, we get $E[S^*(t+1) - S*(t)|Q(t)] \to 0$ w.p. 1, implying that $d_t^{av} z_t \to 0$ w.p. 1. This in turn implies that $z_t \to 0$ w.p. 1, and consequently that $S^*(t) \to n(L)$ w.p. 1. Hence, $P_1^*(t) \to 1$ w.p.1, and thus the $\mathbf{A}$ subautomaton converges to its "optimal" action $a_1^*(t)$ w.p.1. **Q.E.D.**

Theorem 3: In every stationary random environment, the HDPN algorithm is ε-optimal. In other words, given $\varepsilon > 0$ and $\delta > 0$, there exists a $n_0 < \infty$ and a $t_0 < \infty$ such that for all $t > t_0$ and $n > n_0$, $\mathbf{Pr}\{|p_b(t) - 1| < \varepsilon\} > 1 - \delta$.

Proof: Define as $\mathbf{u}$ the difference between the two highest reward probabilities. d_b (the highest reward probability) is unique; thus, there exists a $\mathbf{u} > 0$ such that $d_b - \mathbf{u} \geq d_i$ for all $i \neq b$. By the weak law of large numbers, it can be shown that for a given $\delta > 0$, there exists an $M < \infty$, such that if each action a_i is chosen at least M times, we get the following equation:

$$\mathbf{Pr}\{|d_i'(t) - d_i| < \mathbf{u}/2\} > 1 - \delta. \tag{11}$$

Since M is a constant for each stationary environment, by Theorem 1 it is derived that there exists a time instant $t_0 < \infty$ and a resolution parameter $n_0 < \infty$ such that under the HDPN learning algorithm, for all $n > n_0$ and $t > t_0$, we get the following equation:

$$\mathbf{Pr}\{R_i^i > M\} \geq 1 - \delta. \tag{12}$$

By relation (11), it is derived that for all $j \neq b$, we get the following equation:

$$\mathbf{Pr}\{d_b'(t) > d_j'(t)\} > 1 - \delta, \quad \text{for all } t \text{ such that } \min_i\{R_t^i\} > M. \tag{13}$$

By (12), we can find n_0 and t_0 such that $\mathbf{Pr}\{\min_i\{R_t^i\} > M\} > 1 - \delta$ for all $n > n_0$ and $t > t_0$. By combining this result with (13) and Theorem 2, Theorem 3 follows in a straightforward manner.

REFERENCES

[1] M. A. L. Thathachar and B. J. Oommen, "Discretized reward inaction learning automata," *J. Cybernetics Inform. Sci.*, pp. 24–29, Spring 1979.

[2] B. J. Oommen, "Absorbing and ergodic discretized two-action learning automata," *IEEE Trans. Syst., Man, Cybernetics*, vol. SMC-16, no. 2, pp. 282–293, Mar./Apr. 1986.

[3] B. J. Oommen and J. P. R. Christensen, "Epsilon-optimal discretized linear reward-penalty learning automata," *IEEE Trans. Syst., Man, Cybernetics*, vol. SMC-18, no. 3, pp. 451–458, May/June 1988.

[4] M. A. L. Thathachar and K. R. Ramakrishnan, "A hierarchical system of learning automata," *IEEE Trans. Syst., Man, Cybernetics*, vol. SMC-11, no. 3, pp. 236–241, Mar. 1981.

[5] M. A. L. Thathachar and P. S. Sastry, "A hierarchical system of learning automata that can learn the globally optimal path," *Inform. Sci.*, vol. 42, pp. 143–166, July 1987.

[6] K. S. Narendra and M. A. L. Thathachar, *Learning automata: An introduction.* Englewood Cliffs, NJ: Prentice-Hall, 1989.

[7] B. J. Oommen and J. K. Lanctot, "Discretized pursuit learning automata," *IEEE Trans. Syst., Man, Cybernetics*, vol. SMC-20, no. 4, pp. 931–938, July/Aug. 1990.

[8] M. A. L. Thathachar and P. S. Sastry, "A new approach to the design of reinforcement schemes for learning automata," *IEEE Trans. Syst., Man, Cybernetics*, vol. SMC-15, no. 1, pp. 168–175, Jan./Feb. 1985.

[9] H. G. Tucker, *A Graduate Course in Probability.* New York: Academic Press, 1967.