The Hierarchical Continuous Pursuit Learning Automation for *Large* Numbers of Actions

Anis Yazidi[⋆], Xuan Zhang[⋆⋆], Lei Jiao[⋆⋆⋆], and B. John Oommen[†]

Abstract. Although the field of Learning Automata (LA) has made significant progress in the last four decades, the LA-based methods to tackle problems involving environments with a large number of actions are, in reality, relatively unresolved. The extension of the traditional LA (fixed structure, variable structure, discretized, and pursuit) to problems within this domain cannot be easily established when the number of actions is very large. This is because the dimensionality of the action probability vector is correspondingly large, and consequently, most components of the vector will, after a relatively short time, have values that are *smaller* than the machine accuracy permits, *implying that they will never be chosen*. This paper pioneers a solution that extends the continuous pursuit paradigm to such *large*-actioned problem domains. The beauty of the solution is that it is hierarchical, where all the actions offered by the environment reside as leaves of the hierarchy. Further, at every level, we merely require a *two*-action LA which automatically resolves the problem of dealing with arbitrarily small action probabilities. Additionally, since all the LA invoke the pursuit paradigm, the best action at every level trickles up towards the root. Thus, by invoking the property of the "max" operator, in which, the maximum of numerous maxima is the overall maximum, the hierarchy of LA converges to the optimal action. Apart from reporting the theoretical properties of the scheme, the paper contains extensive experimental results which demonstrate the power of the scheme and its computational advantages. As far as we know, there are no comparable results in the field of LA.

Keywords : *Learning Automata (LA), Pursuit LA, Estimator-based LA, Hierarchical LA, LA with large number of actions.*

⋆ Author's status: *Associate Professor*. This author can be contacted at: Oslo and Akershus University College, Department of Computer Science, Pilestredet 35, Oslo, Norway. E-mail: anis.yazidi@hioa.no.

⋆⋆ This author can be contacted at: Centre for Artificial Intelligence Research, University of Agder, Grimstad, Norway. E-mail: xuan.z.jiao@gmail.com. This author is also a *Data Analyst* in Confirmit AS, Norway.

⋆⋆⋆ Author's status: *Associate Professor*. This author can be contacted at: Department of ICT, University of Agder, Grimstad, Norway. E-mail: lei.jiao@uia.no.

† *Chancellor's Professor*; *Fellow: IEEE* and *Fellow: IAPR*. This author can be contacted at: School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6. This author is also an *Adjunct Professor* with the University of Agder in Grimstad, Norway. E-mail address: oommen@scs.carleton.ca. The work of this author was partially supported by NSERC, the Natural Sciences and Engineering Council of Canada.

1 Introduction

This paper deals with the well-trodden field of Learning automata (LA)[1]. For decades, this field, initiated by Tsetlin [10], has been studied as a typical model for learning in random environments, and has served as the precursor for the area of reinforcement learning. Unlike other fields of Artificial Intelligence (AI), an LA, by definition, operates in random environments, where the "Teacher" can respond differently and randomly, for the same query, at different time instances. More specifically, an LA is an adaptive decision-making unit that learns the optimal action from among a set of actions offered by the Environment that it operates in. Without loss of generality, to render the problem non-trivial, the Environment is stochastic. At each iteration (or time step), the LA selects one action and communicates it to the Environment. This, in turn, *stochastically* triggers either a reward or a penalty as a response from the Environment. Based on the response and the knowledge acquired in the past iterations, the LA, either deterministically or stochastically, adjusts its action selection strategy. This is done so as to make a "wiser" decision in the next iteration. Thus the LA, even though it lacks a complete knowledge about the Environment, is able to learn through repeated interactions with the Environment, and adapts itself, or "converges", to the optimal decision.

Although LA have been studied extensively [1,13–15,17] and been applied in many fields [4,9], designing LA when the number of actions involved, R, is large is extremely complex. The solution that we propose in this paper attempts to resolve this problem.

1.1 Contributions of the Paper

The contributions of this paper are the following:

1. We propose a hierarchical LA strategy which superimposes the learning process on a tree structure. Unlike the traditional hierarchical schemes, we do not resort to Fixed Structure Stochastic Automata (FSSA) or Variable Structure Stochastic Automata (VSSA) to achieve the learning.
2. We propose a novel learning process that involves multi-level two action Continuous Pursuit Algorithm (CPA) machines. The estimation and interaction with the real-world environment occur *only at the leaf level*.
3. We propose the process of trickling-up the estimates and accomplishing the learning by *only* invoking learning between a node and its sibling.
4. The scheme that we have proposed is novel in that it never involves action probabilities that are below machine accuracy. It also involves estimates whose accuracies can easily be attained.
5. The scheme that we have proposed is ε-optimal in all random environments [12].

[1] In the interest of brevity, we assume that the reader is fairly well-versed in the fundamental concepts of LA and their convergence properties. The review here is thus necessarily brief, although the intent is that it should also be comprehensive. However, excellent surveys of the field can be found in [5, 8, 9]. Also, due to the space limitations, the theoretical results about our new scheme are omitted here. They are included in [12].

6. The speed of the proposed scheme is *many* times faster than that of all the LA reported in the literature. It is thus the fastest and most accurate reported LA for environments with a large number of actions. As far as we know, no experiments have ever been done in the field of LA for environments when the number of actions was so large, and in that sense, this is truly a pioneering and ground-breaking venture, clearly proving the power of the scheme!

2 The HCPA LA

2.1 Rationale for Our Solution

The philosophy motivating our new scheme resorts to superimposing the actions onto a binary tree[2], in which, the leaves are the actual actions themselves. Further, each internal node represents the best action in the *entire subtree* below that node. By performing comparisons between the actions in a pairwise manner, i.e., at the leaves of the tree, only the superior actions are trickled up towards the root. By doing this, one always deals with 2-action LA. Here, however, unlike the work of previous researchers [2], we do not resort to FSSA or traditional VSSA, to differentiate between the various pairs of actions at the leaves. Rather, we shall use the 2-action continuous pursuit LA [16]. Since $R = 2$ at every level, the number of iterations required to achieve the estimation is considerably less. Further, the estimation that is achieved at the leaf level, is all that is required for the entire tree – no estimation operations are required at the internal nodes.

A notable attempt to devise hierarchical LA is due to Papadimitriou [7]. Before we comment on this work, we mention that the Pursuit concept can be used in a Continuous or Discretized paradigm, and that the action probabilities can be changed on Reward-Penalty (*RP*), Reward-Inaction (*RI*) and Inaction-Penalty (*IP*) scenarios. Consequently, we would have six Pursuit variants: CP_{RP}, DP_{RP}, CP_{RI}, DP_{RI}, CP_{IP} and DP_{IP}, and of these, Agache and Oommen [6] showed that the DP_{RI} is the most superior one. The author of [7] has precisely used this machine, and this is commendable. The differences between that work and the work that we have done here is, however, significant. First of all, this lies is in the way that we have modeled the tree along which the actions have been placed. Secondly, the strategy by which we have trickled up the "maximum" estimate at every node is quite unique and novel, and it does not require us to probe (query) the environment at every time instant, implying that these interactions with the Environment are only at the leaves. All of these lead to the superiority of our scheme over the recorded ones, demonstrated for experiments done for a much larger set than what has been reported in the literature[3]!

[2] The tree is assumed to be binary only for the sake of convenience. In a more general setting, each node may have, for example, three children.

[3] The author of [7] tested his scheme for a maximum of 64 actions. It was not possible to do a fair comparison between our scheme and the work done in [7]. This is because the author of [7] did not report the size of the ensemble of experiments that he conducted. In our case, the ensemble was of size 400, and we sought for the best parameter that yielded "absolute" convergence - i.e., convergence in every single experiment. However, the author of [7] should be given fair credit because of his result being the first reported hierarchical Pursuit-based LA strategy!

2.2 Construction of the Hierarchy

The search space for the binary tree alluded to above is constructed as follows. First of all, the hierarchy is organized as a balanced full[4] binary tree with maximal depth K. For the sake of convenience and in the interest of mathematical formalism, we will use the same notation adopted in [3, 11], and index the nodes using both their depth in the tree and their relative order with respect to the nodes located at the same tree depth. The details of the hierarchy as described as follows.

1. **Root node**: The LA at the root of the hierarchy is the one at depth 0.
2. **The various LA**: At each node, we invoke a 2-action LA $\mathcal{A}$, whose actions are cited as 0 and 1.
3. **LA activations for K levels: from 0 to K-1**:
   - **The different LA at depth** k: The LA $j \in \{1,...,2^k\}$ at depth k, is referred to as $\mathcal{A}_{\{k,j\}}$, where $0 \leq k < K-1$, and it has two actions $\alpha_{\{k+1,2j-1\}}$ and $\alpha_{\{k+1,2j\}}$.
     - Whenever the action $\alpha_{\{k+1,2j-1\}}$ is chosen, the LA $\mathcal{A}_{\{k+1,2j-1\}}$ is activated.
     - Whenever the action $\alpha_{\{k+1,2j\}}$ is chosen, the LA $\mathcal{A}_{\{k+1,2j\}}$ is activated.
     - We can informally say that $\mathcal{A}_{\{k+1,2j-1\}}$ and $\mathcal{A}_{\{k+1,2j\}}$ are the *Left Child* and *Right Child* of the parent LA $\mathcal{A}_{\{k,j\}}$ respectively.
   - **The LA at depth** $K-1$: The LA at depth $K-1$ (i.e., at the level *just* above the leaves) is responsible for choosing the action from the stochastic environment.
     - This LA has two actions $\alpha_{\{K,2j-1\}}$ and $\alpha_{\{K,2j\}}$.
     - At this level, there are 2^K actions: $\alpha_{\{K,j\}}$ where $j \in \{1,...,2^K\}$ at depth K.
     - Observe that $\alpha_{\{K,j\}}$ is attached to (or associated with) its "parent LA" $\mathcal{A}_{\{K-1,\lceil j/2 \rceil\}}$.
4. **At level K**: Finally, at depth K, i.e., at the maximal depth of the tree, the nodes do not have children.

2.3 The Proposed Solution

At the bottom-most level, i.e., the level of the leaves, we invoke a two-action CPA to determine which is the superior action between two actions that are siblings at this level. To do this, we merely maintain running estimates of the reward probabilities of *these* two actions, and using this two-dimensional estimate vector and the corresponding two-action probability vector, the updating is achieved. The larger of these estimates is trickled to their common parent, and this estimate is now compared with the corresponding reward probability estimate of *its* sibling whose value was obtained from *its* children. This process is now recursively repeated, using the estimate of the reward probability at this level and the probability vector at this level, whence the updates are performed. The same process continues up the tree to the root itself.

[4] If the number of actions is less than 2^K, one can always add dummy actions whose reward probabilities are zero.

2.4 The Algorithm of the Proposed Solution

Notation and Definitions The notation that we shall use is as follows:

- The 2^K actions that interact with the Environment are elements from the set $\{\alpha_{\{K,1\}}, \ldots, \alpha_{\{K,2^K\}}\}$. Further, the actions $\{\alpha_{\{K,2j-1\}}, \alpha_{\{K,2j\}}\}$ are the two only actions that can be selected by the LA at level $K-1$, namely $\mathcal{A}_{\{K-1,j\}}$.
- Each LA $j \in \{1,...,2^k\}$ at depth k, called $\mathcal{A}_{\{k,j\}}$, where $0 \leq k \leq K-1$ has two actions, namely, $\alpha_{\{k+1,2j-1\}}$ and $\alpha_{\{k+1,2j\}}$.
- $P_{\{k,j\}} = [p_{\{k+1,2j-1\}}, p_{\{k+1,2j\}}]^T$ is the action probability vector of LA $\mathcal{A}_{\{k,j\}}$, where $0 \leq k \leq K-1$.

Begin Algorithm HCPA

Parameters:
λ: The learning parameter, where $0 < \lambda < 1$, where λ is close to zero.
$u_{\{K,2j-1\}}, u_{\{K,2j\}}$: The number of times $\alpha_{\{K,2j-1\}}, \alpha_{\{K,2j\}}$ have been rewarded *when* it has been selected.
$v_{\{K,2j-1\}}, v_{\{K,2j\}}$: The number of times $\alpha_{\{K,2j-1\}}, \alpha_{\{K,2j\}}$, has actually been selected.
$\hat{d}_{\{K,2j-1\}}, \hat{d}_{\{K,2j\}}$: The estimate of the reward probabilities of $d_{\{K,2j-1\}}, d_{\{K,2j\}}$, computed as: $\hat{d}_{\{K,2j-1\}} = \frac{u_{\{K,2j-1\}}}{v_{\{K,2j-1\}}}, \hat{d}_{\{K,2j\}} = \frac{u_{\{K,2j\}}}{v_{\{K,2j\}}}$.
$\hat{D}$ is the vector of the estimates $\{\hat{d}\}$.
m: The index of the optimal action.
h: The index of the greatest element of $\hat{D}$.
R: The response from the Environment, where $R = 0$ corresponds to a Reward, and $R = 1$ to a Penalty.
T: A Threshold, where $T \geq 1 - \varepsilon$.
Initialization: Traditional Pursuit algorithms require that we choose each action a few times to initialize the estimates of the reward probabilities. This step is really not so crucial and so we have avoided it and assumed that the estimate of the reward probabilities are initialized to 0.5.
Initialization:
$t = 0$
For i= 1 to 2^K Do:
  $u_{\{K,i\}}(0) = 1$
  $v_{\{K,i\}}(0) = 2$
  $\hat{d}_{\{K,i\}}(0) = \frac{u_{\{K,i\}}(0)}{v_{\{K,i\}}(0)}$
EndFor
Loop

1. $0 \leq k < K-1$: **Levels 0 to** $K-1$
   - LA $\mathcal{A}_{\{0,1\}}$ selects an action by randomly sampling as per the action probability vector $[p_{\{1,1\}}(t), p_{\{1,2\}}(t)]$.
   - Let $j_1(t)$ be the index of the chosen action where $j_1(t) \in \{1,2\}$.
   - The next LA is activated $\mathcal{A}_{\{1,j_1(t)\}}$ which in turn chooses an action and activates the next LA at level '2'.
   - The procedure continues recursively until LA at level $K-1$.
   - Let $\mathcal{A}_{\{k,j_k(t)\}}$ be the set of activated LA, where j_k denotes the activated LA at level k.

2. $k = K$: **Level** K
   - Update $\hat{D}_{\{K,j_K(t)\}}$ based on the response from the Environment at the leaf level, K:
     $$u_{\{K,j_K(t)\}}(t) = u_{\{K,j_K(t)\}}(t-1) + (1-R(t))$$
     $$v_{\{K,j_K(t)\}}(t) = v_{\{K,j_K(t)\}}(t-1) + 1$$
     $$\hat{d}_{\{K,j_K(t)\}}(t) = \frac{u_{\{K,j_K(t)\}}(t)}{v_{\{K,j_K(t)\}}(t)}.$$
   - For all other "leaf actions", where $j \in \{1,...,2^k\}$ and $j \neq j_K(t)$,
     $$u_{\{K,j\}}(t) = u_{\{K,j\}}(t-1)$$
     $$v_{\{K,j\}}(t) = v_{\{K,j\}}(t-1)$$
     $$\hat{d}_{\{K,j\}}(t) = \frac{u_{\{K,j\}}(t)}{v_{\{K,j\}}(t)}.$$
3. Define the reward estimate for all other actions along the path from the root, $0 < k < K-1$ in a recursive manner[5], where the LA at any one level inherits the feedback from the LA at the next level:
   $$\hat{d}_{\{k,j\}}(t) = max(\hat{d}_{\{k+1,2j-1\}}(t), \hat{d}_{\{k+1,2j\}}(t)).$$
4. Perform the probability updating for the corresponding vectors as follows:
   - By definition, each LA $j \in \{1,...,2^k\}$ at depth k, referred to as $\mathcal{A}_{\{k,j\}}$, where $0 \leq k \leq K-1$, has two actions $\alpha_{\{k+1,2j-1\}}$ and $\alpha_{\{k+1,2j\}}$. Let $j^h(t) \in \{2j-1, 2j\}$ be the larger of the elements between $\hat{d}_{\{k+1,2j-1\}}(t)$ and $\hat{d}_{\{k+1,2j\}}(t)$.
   - Let $\overline{j^h(t)} = \{2j-1, 2j\} \setminus j^h(t)$ be the opposite action, i.e., the one that has the lower reward estimate.
   - Update $p_{\{k,j^h(t)\}}$ and $p_{\{k,\overline{j^h(t)}\}}$ using the estimates $\hat{d}_{\{k+1,2j-1\}}(t)$ and $\hat{d}_{\{k+1,2j\}}(t)$ as:
     If $R(t) = 0$ **Then**
     $$p_{\{k,\overline{j^h(t)}\}}(t+1) = (1-\lambda) p_{\{k,\overline{j^h(t)}\}}$$
     $$p_{\{k,j^h(t)\}}(t+1) = 1 - p_{\{k,\overline{j^h(t)}\}}(t+1).$$
     Else
     $$p_{\{k,\overline{j^h(t)}\}}(t+1) = p_{\{k,\overline{j^h(t)}\}}(t)$$
     $$p_{\{k,j^h(t)\}}(t+1) = p_{\{k,j^h(t)\}}(t).$$
     EndIf
   - For each $\mathcal{A}_{\{k,j\}}$, if either of its action probabilities $p_{\{k+1,2j-1\}}$ and $p_{\{k+1,2j\}}$ surpasses a threshold T, where T is a positive number that is close to unity, the action probabilities for this LA will stop updating, with its larger action probability jumping to unity.
5. $t = t+1$

EndLoop
End Algorithm HCPA

The HCPA scheme proposed and described above has been shown to be ε-optimal in all random environments. The proofs are quite deep and intricate. However, due to the space limitations, these theoretical results are omitted here. They are included in [12].

[5] More specifically, the LA at level $K-2$, inherit the feedback from the LA at level $K-1$ as:
$$\hat{d}_{\{K-2,j\}}(t) = max(\hat{d}_{\{K-1,2j-1\}}(t), \hat{d}_{\{K-1,2j\}}(t))$$
and so on. As a consequence, notice that *at every level*, the reward vector estimates of the actions of every LA, are composed of the respective *maxima* of the rewards of all the actions of the *entire* subtrees rooted at their children.

3 Experimental Results

To evaluate the performance of the LA-based schemes, we carried out extensive simulations for environments with a "large" number of actions, where the total number of actions was set to various values. The main aspect that we intended to demonstrate was that if the learning problem was tackled using traditional VSSA, the convergence would be both less accurate and very slow. The reason for this, as mentioned earlier, is that if the number of actions is large, many of the action probabilities would be small, implying that these would be chosen seldom. Thus, even if we invoked estimator-based LA, it would be unreasonable to assume that each action would be chosen "a large number of times". Further, the estimates would be correspondingly inaccurate. The HCPA resolves both of these issues.

The simulations that we conducted were intended to capture two important metrics, namely, the accuracy of the convergence of HCPA, and its speed of the convergence. Our goal was also to compare its convergence with the existing LA.

3.1 The Data Sets for the Environment

The benchmark datasets reported in the existing literature had at most ten actions. In the absence of established benchmarks for larger numbers of actions, we have designed a set of Environments which can be used as benchmarks by other researchers. First of all, we determined the number of actions involved in the learning problem. To render the problem non-trivial, the total numbers of actions was initially configured to be 16, 32 and 64. Once the number of actions was set, the actual reward probabilities associated with the different actions were uniformly distributed in the interval between zero and unity. Understandably, the difficulty the Environment increased with the the number of actions. The reward probabilities associated with the configurations for 16 and 32 actions are the first 16 and 32 elements in Table 1, respectively. The reward probabilities of the configuration with 64 actions constitute the entire set given in Table 1.

3.2 Convergence of the HCPA Algorithm

If λ is sufficiently small, the HCPA will converge to the action with the maximum reward probability. To observe the convergence of the algorithm with a minimum number of iterations, our task was to determine the optimal value for λ for different configurations. The optimal λ value is the maximum λ value that will make the LA to *consistently* converge to the correct action. Obviously, for different configurations for the Environment, the value for optimal λ would vary. In this simulation, to find the optimal λ, we decreased the value of λ until we reached the one that provided the LA for the first 200 consecutive occurrences of convergence to the correct action.

Based on our simulations, for the configuration with 64 actions, the optimal λ was 0.000051. In other words, with this value of $\lambda \leq 0.000051$ system would consistently converge accurately. Similarly, the optimal values for λ for the configurations for 32 and 16 actions were 0.00085 and 0.0065 respectively. Understandably, the values of λ have an increasing trend when the environment becomes less challenging.

Table 1: This table lists the reward probability of the 64 actions in our experiments. The reward probabilities for 16 and 32 actions are the corresponding 16 and 32 entries in the table, respectively.

A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}
0.3934	0.9902	0.4883	0.5768	0.2023	0.2390	0.5887	0.8894	0.0333	0.4323	0.6926	0.3474
A_{13}	A_{14}	A_{15}	A_{16}	A_{17}	A_{18}	A_{19}	A_{20}	A_{21}	A_{22}	A_{23}	A_{24}
0.6152	0.0900	0.0850	0.5652	0.7362	0.7603	0.5142	0.2273	0.6080	0.4791	0.9339	0.3808
A_{25}	A_{26}	A_{27}	A_{28}	A_{29}	A_{30}	A_{31}	A_{32}	A_{33}	A_{34}	A_{35}	A_{36}
0.02152	0.2399	0.7509	0.8773	0.4962	0.5649	0.9202	0.1335	0.6214	0.9777	0.4232	0.02773
A_{37}	A_{38}	A_{39}	A_{40}	A_{41}	A_{42}	A_{43}	A_{44}	A_{45}	A_{46}	A_{47}	A_{48}
0.1255	0.5650	0.1660	0.0148	0.0970	0.1319	0.1738	0.8901	0.3511	0.8945	0.6133	0.4813
A_{49}	A_{50}	A_{51}	A_{52}	A_{53}	A_{54}	A_{55}	A_{56}	A_{57}	A_{58}	A_{59}	A_{60}
0.2413	0.1714	0.8512	0.9791	0.7443	0.3469	0.8707	0.3863	0.4763	0.4446	0.9617	0.0329
A_{61}	A_{62}	A_{63}	A_{64}								
0.5004	0.3784	0.6553	0.9737								

3.3 Average Convergence Iterations

To illustrate the average number of iterations before convergence, we present the simulation results of the experiments[6] in Tables 2. The standard deviation of the iterations are also included. To compare the HCPA with existing approaches, we include the simulation results for the L_{R-I} and CPA machines in the same environment. The λ values utilized in the HCPA are the ones shown in Section 3.2 while the ones in the CPA are the optimal values found based on the same approach explained in Section 3.2.

For each replication in HCPA, we register the number of iterations when all the LAs along the correct path had converged to the action probabilities which are greater than or equal to 0.99. Similarly, for each trial for the CPA and the L_{R-I}, we record the number of iterations when the LA had converged to the correct action with an action probability greater than or equal to 0.99. All the results presented in the table have been averaged over an ensemble of 400 independent replications using the optimal λ determined above.

Table 2: The simulation results obtained for various environments with different numbers of actions.

Number of Actions	**16**		**32**		**64**	
Parameters	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
HCPA	**904.5**	**103.6**	**6,812.3**	**614.6**	**115,295.5**	**11,346.2**
CPA	**1,584.2**	**62.3**	**7,260.0**	**529.1**	**156,616.3**	**6,985.0**
L_{R-I}	**3,920.8**	**1,629.2**	**28,618.2**	**7,911.3**	**644,234.0**	**20,0625.4**

[6] The experiments have been done for various randomly-generated environments. In the interest of brevity, we merely report the results from one such setting. This was representative of the results obtained for other settings.

From Table 2, we can clearly see that HCPA outperforms CPA and L_{R-I} in general, especially when the number of actions is large. Thus, for example, for the 64-action environment, the L_{R-I} required 644,234 iterations. The HCPA required less than 18% of the number of iterations, namely 115,295. These results are typical. This confirms the efficiency of the hierarchical structure when the number of actions increases.

3.4 Environment with 128 actions

The HCPA was also tested on environment with 128 actions, and as mentioned earlier, the testing of LA in environments with such a large number of actions is pioneering – it has been unreported in the literature. Rather than list the reward probabilities, we have plotted them in Figures 1.

Fig. 1: An example of an 128-action Environment.

In the case of the first environment plotted in Figure 1, the L_{R-I} required 734,474 steps for absolute convergence for an ensemble of 400 trials. The CPA, on the other hand, required 543,529 steps - which represented a decrease of about 26%. Astonishingly, the HCPA needed only 266,257 steps. This implied an advantage of about 51% over the CPA and of almost 64% over the L_{R-I}. One can clearly see the advantage of the HCPA over the state-of-the-art.

4 Conclusions

In this paper, we have pioneered a new paradigm for designing and implementing Learning Automata (LA) when the number of actions is large. Learning in environments of this type is particularly hard because the dimensionality of the action probability vector is correspondingly large, and consequently, most components of the vector will, after a relatively short time, have values that are *smaller* than the machine accuracy, *implying that they will never be chosen*. This means that the traditional LA will be sluggish and inaccurate, and it would be unreasonable to assume that each action would be chosen "a large number of times" if we invoked estimator-based LA. In this paper, we have pioneered a solution that extends the Continuous Pursuit Algorithm's (CPA's) paradigm to such *large*-actioned problem domains. The salient feature of our new solution is that it is hierarchical, where all the actions offered by the environment reside as leaves of the hierarchy. Further, at every level, we merely require a *two*-action LA which automatically resolves the problem of dealing with arbitrarily small action probabilities. Most importantly, since all the LA invoke the pursuit paradigm, the best action at every level trickles up towards the root. Thus, by invoking the property of the "max" operator, in which, the maximum of numerous maxima is the overall maximum, the hierarchy of LA converges to the optimal action. The paper also reported experimental results that demonstrated the power of the scheme and its computational advantages.

References

1. Agache, M., Oommen, B.J.: Generalized pursuit learning schemes: new families of continuous and discretized learning automata. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 32(6), 738–749 (2002)
2. Baba, N., Mogami, Y.: A new learning algorithm for the hierarchical structure learning automata operating in the nonstationary S-model random environment. IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics 32(6), 750–758 (2002)
3. Granmo, O.C., Oommen, B.J.: Solving stochastic nonlinear resource allocation problems using a hierarchy of twofold resource allocation automata. IEEE Transactions on Computers 59, 545–560 (2009)
4. Jiao, L., Zhang, X., Oommen, B.J., Granmo, O.C.: Optimizing channel selection for cognitive radio networks using a distributed bayesian learning automata-based approach. Applied Intelligence 44(2), 307–321 (Mar 2016)
5. Obaidat, M.S., Papadimitriou, G.I., Pomportsis, A.S.: Learning automata: Theory, paradigms, and applications. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 32(6), 706–709 (December 2002)
6. Oommen, B.J., Agache, M.: Continuous and discretized pursuit learning schemes: various algorithms and their comparison. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 31(3), 277–287 (2001)
7. Papadimitriou, G.I.: Hierarchical discretized pursuit nonlinear learning automata with rapid convergence and high accuracy. IEEE Transactions on Knowledge and Data Engineering 6(4), 654–659 (1994)
8. Poznyak, A.S., Najim, K.: Learning Automata and Stochastic Optimization. Springer-Verlag, Berlin (1997)
9. Thathacha, M.A.L., Sastry, P.S.: Networks of Learning Automata: Techniques for Online Stochastic Optimization. Kluwer Academic Publishers (2004)
10. Tsetlin, M.L.: Finite automata and the modeling of the simplest forms of behavior. Uspekhi Matem Nauk 8, 1–26 (1963)
11. Yazidi, A., Granmo, O.C., Oommen, B.J., Goodwin, M.: A novel strategy for solving the stochastic point location problem using a hierarchical searching scheme. IEEE transactions on cybernetics 44(11), 2202–2220 (2014)
12. Yazidi, A., Zhang, X., Jiao, L. and Oommen, B. J.: The Hierarchical Continuous Pursuit Learning Automation: A Novel Scheme for Environments with *Large* Numbers of Actions. Unabridged version of this paper (2018)
13. Zhang, X., Granmo, O.C., Oommen, B.J.: The Bayesian pursuit algorithm: A new family of estimator learning automata. In: Proceedings of IEA-AIE 2011. pp. 608–620. Springer, New York, USA (Jun 2011)
14. Zhang, X., Granmo, O.C., Oommen, B.J.: Discretized Bayesian pursuit - a new scheme for reinforcement learning. In: Proceedings of IEA-AIE 2012. pp. 784–793. Dalian, China (Jun 2012)
15. Zhang, X., Granmo, O.C., Oommen, B.J.: On incorporating the paradigms of discretization and Bayesian estimation to create a new family of pursuit learning automata. Applied Intelligence 39, 782–792 (2013)
16. Zhang, X., Granmo, O.C., Oommen, B.J., Jiao, L.: A formal proof of the ε-optimality of absorbing continuous pursuit algorithms using the theory of regular functions. Applied Intelligence 41(3), 974–985 (2014)
17. Zhang, X., Oommen, B.J., Granmo, O.C.: The design of absorbing Bayesian pursuit algorithms and the formal analyses of their ε-optimality. Pattern Analysis and Applications 20(3), 797–808 (Aug 2017), https://doi.org/10.1007/s10044-016-0535-1