The Hierarchical Discrete Learning Automaton Suitable for Environments with *Many* Actions and *High* Accuracy Requirements

Rebekka Olsson Omslandseter[1], Lei Jiao [1], Xuan Zhang [2], Anis Yazidi [3], and B. John Oommen [1,4]

[1] Dept. of Information and Communication Technology, University of Agder, 4879, Grimstad, Norway, {rebekka.o.omslandseter, lei.jiao}@uia.no
[2] Norwegian Research Centre (NORCE), 4879, Grimstad, Norway
[3] Oslo Metropolitan University, 0167, Oslo, Norway
[4] Carleton University, Ottawa, Canada

Abstract. Since its early beginning, the paradigm of Learning Automata (LA), has attracted much interest. Over the last decades, new concepts and various improvements have been introduced to increase the LA's speed and accuracy, including employing probability updating functions, discretizing the probability space, and implementing the "Pursuit" concept. The concept of incorporating "structure" into the ordering of the LA's actions is one of the latest advancements to the field, leading to the ϵ-optimal Hierarchical Continuous Pursuit LA (HCPA) that has superior performance to other LA variants when the number of actions is *large*. Although the previously proposed HCPA is powerful, its speed has a handicap when the required action probability of an action is approaching unity. The reason for this slow convergence is that the learning parameter operates in a multiplicative manner within the probability space, making the increment of the action probability smaller as its probability becomes close to unity. Therefore, we propose the novel Hierarchical Discrete Learning Automata (HDPA) in this paper, which does not possess the same impediment as the HCPA. The proposed machine infuse the principle of discretization into the action probability vector's updating functionality, where this type of updating is invoked recursively at every depth within a hierarchical tree structure and we pursue the best estimated action in all iterations through utilization of the Estimator phenomenon. The proposed machine is ϵ-optimal, and our experimental results demonstrate that the number of iterations required before convergence is significantly reduced for the HDPA, when compared with the HCPA.

Keywords: Reinforcement Learning · Learning Automata · Hierarchical Discrete Pursuit LA.

1 Introduction

In the field of Learning Automata (LA), non-human agents, implemented through computer programs, find solutions to problems of stochastic nature through the concept of learning. One of the main advantages of this type of Machine Learning (ML) is the scheme's ability to operate adaptively. The paradigm is based on a learning agent, or Learning Automaton, referred to as a LA, that interacts with a teacher, referred to as the *Environment* [4]. The LA has several actions that often correspond to the different solutions for the given problem. Through selecting actions and getting feedback from the Environment, the LA learns which action (behavior) results in the highest probability of receiving a Reward from the Environment. Although the LA requires feedback from an Environment, it learns in a Semi-Supervised manner. Thus, the LA does not need examples of solutions to learn. They explore different actions and learn via trial-and-error. We can model the Environment in numerous ways. The reader should note that, in this paper, the shortened term LA refers to both the field of Learning Automata and the Learning Automaton according to the context where it appears.

In LA, we evaluate the schemes' performance by the number of iterations needed before convergence and the schemes' accuracy in finding the optimal solution [4]. Improvements that have enhanced the accuracy and speed in LA include using action probability vectors to decide the LA behavior (VSSA), discretizing the probability space of these vectors (Discrete LA), and deploying the "pursuit" concept. In addition, pursuing the most likely action to receive a Reward throughout the LA operation (Estimator-based LA) and ordering the LA in a hierarchical structure (Hierarchical LA) improved the LA's performance further. The structuring of the LA led to the state-of-the-art ϵ-optimal Hierarchical Continuous Pursuit LA (HCPA) [12], which can handle a large number of actions.

Although the HCPA handles a large number of actions, its convergence speed has an impediment when any action probability approaches unity. The reason for this slow convergence is that the learning parameter, and thus the updating of the probabilities operates in a multiplicative manner within the probability space. Consequently, the learning rate decreases as the probability approaches unity. In more detail, as the learning continues, the increment of action probability is less and less, making it more challenging for the LA to converge in the latter phase of learning. This drawback is even more apparent when the criterion for convergence is high, i.e., when high accuracy is required. To solve this problem, we propose the novel ϵ-optimal Hierarchical Discrete Learning Automata (HDPA) scheme in this paper. The HDPA includes all the phenomena mentioned earlier to speed up the convergence when high accuracy is required. The beauty of the HDPA is that the learning speed does not decrease as the learning continues because it operates in a step-wise manner in the probability space. Our simulation results demonstrate that the HDPA has significantly faster convergence for high accuracy requirements. Thus, the HDPA outperforms the state-of-the-art HCPA scheme presented in [12] when the accuracy requirement is high.

Our contributions are summarized as follows:

- We propose the novel HDPA that converges faster than the state-of-the-art HCPA algorithm, when the accuracy requirement is high. The advantages become more pronounced when a large number of actions exist.
- Via extensive simulations, we demonstrate how much the HDPA converged faster than the HCPA for Environments with *many* actions and high accuracy requirements.

2 Related Work

Michael Lvovitch Tsetlin pioneered the field of LA by inventing the Tsetlin Automata (TA) [10], which can be categorized as a Fixed Structure Stochastic Automata (FSSA). The FSSA has a discrete and state-based structure, where the automaton's current state determines its action. The first significant improvement in LA was achieved through the discovery/invention of Variable Structure Stochastic Automata (VSSA), where the action of the LA is selected by randomly sampling an action probability vector. The probability vector is updated through functions according to the Environment's feedback, influencing and changing the behavior of the LA (where this updating functionality can also change over time). While an FSSA needs to move through numerous states before exploring another action, VSSA can possibly explore distinct actions along consecutive iterations, speeding up the exploration of the Environment.

The earlier established variants, the Linear Reward-Penalty (L_{R-P}) scheme, the Linear Reward-Inaction (L_{R-I}) scheme, the Linear Inaction-Penalty (L_{I-P}) scheme, and the Linear Reward-ϵPenalty ($L_{R-\epsilon P}$) scheme in [1] and [4] are all examples of continuous VSSA schemes. The "linear"(L) schemes have such a categorization because the action probabilities are increased in a linear manner. The probabilities of the VSSA can also be increased in a non-linear manner [1, 2, 4]. In mathematical analyses, LA can described through Markov chains, where we have ergodic and absorbing types [1, 8].

The next quantum step in terms of speed and accuracy in LA was achieved through discretizing the action probability space [5]. In traditional VSSA, the action selection probabilities can assume any real value in the interval $[0, 1]$, and the updating is achieved in a multiplicative manner with a learning parameter ($\lambda \in (0, 1)$). The drawback of this continuous approach is its sluggish convergence. As the action probability approach unity, the updating step becomes smaller and smaller, slowing down the algorithm's speed. To address this issue, discretizing the probability space and updating the action probabilities in constant steps was proposed, which significantly improved the convergence speed in LA. The different properties (absorbing and ergodic) of these discretized LA, and their different updating schemes were studied in [5, 6]. In addition, continuous and discretized updating mechanisms with mathematical analyzes are investigated in [7] and [13], respectively.

The invention of Estimator-based Algorithms (EAs) increased the achieved convergence speed of LA even further [9]. The EAs possessed a faster convergence than all earlier variants. These algorithms are based on VSSA, but in addition, estimates of the reward probabilities of the different actions are maintained in a separate vector. These reward estimates are employed in updating the action selection probabilities, where the LA pursues the currently estimated *best action* in terms of the reward estimates (referred to as the Pursuit paradigm). The reward estimates can be found by Maximum Likelihood, or in a Bayesian manner, investigated in [11]. Thereafter, the researchers combined dicretization and the Pursuit paradigm and introduced the family of Discrete Estimator Algorithms (DEAs) [3].

For the above mentioned LA, the convergence becomes challenging when the number of possible actions is large. Understandably, for VSSA, the action probability vector has a dimension of R (for R actions) and its elements sum up to 1. When R is large, many of the action probabilities can have very small values and may not even be chosen, thus rendering the principle behind VSSA to be void. Inclusion of structure into the field of LA solved this problem, and the HCPA constitutes the state-of-the-art [12]. Although HCPA solves the problem to a certain extent, it is always valuable if we can improve the convergence speed of the algorithm without satisfying the accuracy, leading to the proposed algorithm in this paper.

3 The Proposed Algorithm

The concept of the HDPA is to utilize VSSA, discretizing the probability space, structuring the LA instances in a hierarchical tree structure and incorporating the Estimator concept. In more detail, we organize a set of Discrete Pursuit Automata (DPA) instances in a tree structure, where each instance has a set of actions corresponding to the possible paths down the tree structure from that automaton. The probabilities of these actions are maintained through vectors that are updated in a discretized manner. At the bottom level of the tree, we have the actions that directly interact with the Environment. The HDPA maintains reward estimates of all the actions throughout the tree structure, and we pursue the action with the currently best reward estimate in all iterations according to the pursuit paradigm. The reader should note that, in reality, the reward estimates are only necessary for the actions at the leaf level. We utilize the reward estimates in this way, because the proof of the algorithm's convergence needs the reward estimates along the path[5]. A more detailed explanation of the algorithm is given in what follows.

3.1 The Structure of the HDPA

For ease of explanation of the HDPA in this paper, we utilize 2-action DP_{RI} instances as the LA in the tree structure. The reason for using Reward-Penalty

[5] A formal proof of the HDPA's convergence will be given in an extended version of this paper.

Inaction LA is that they have demonstrated better performance than other configurations [12]. Furthermore, we use 2-actions automatons in our explanations. Therefore, we can model the HDPA as a balanced full binary tree for a problem with 2^K actions, where K is the maximum depth of the tree. The number of actions per LA instance can be changed to another configuration. However, the reader should remember that one of the main reasons for organizing the actions in a tree structure is to mitigate a large action probability vector because of its inferior performance [12]. Therefore, the number of actions in the LA instances should, in any case, be limited. The HDPA in these explanations is, thus, configured to handle 2^K original actions. If the number of original actions is not 2^K, we consider the nearest power of 2 above the number of original actions and configure the excess number of actions with zero Reward probability. To continue our explanations in greater detail, we further formalize the levels in the tree structure as follows:

- **Hierarchical tree structure**: The depth of the tree is indexed by parameter k, $k \in \{0, ..., K\}$. For each level of depth, the number of nodes is indexed by $j \in \{1, ..., 2^k\}$. Note that k and j are also employed to index different LA and actions with their corresponding ranges of definition.
- **The various LA**: The LA $j \in \{1, ..., 2^k\}$ at depth k, is referred to as $\mathcal{A}_{\{k,j\}}$, where $k \in \{0, ..., K-1\}$. The LA at the root is the one at depth 0.
- **The LA at depths from 0 to $K-1$ ($0 \le k < K-1$)**:
  - Each of the LA, $\mathcal{A}_{\{k,j\}}$, has two actions, denoted by $\alpha_{\{k+1,2j-1\}}$ and $\alpha_{\{k+1,2j\}}$, respectively.
  - Whenever the action $\alpha_{\{k+1,2j-1\}}$ is chosen, the LA, $\mathcal{A}_{\{k+1,2j-1\}}$, at the next level is activated.
  - Whenever the action $\alpha_{\{k+1,2j\}}$ is chosen, the LA, $\mathcal{A}_{\{k+1,2j\}}$, at the next level is activated.
- **The LA at depth $K-1$ ($k = K-1$)**: The LA at depth $K-1$ select the actual actions to interact with the Environment.
  - All of the LA at depth $K-1$ have two possible actions each, referred to as $\alpha_{\{K,2j-1\}}$ and $\alpha_{\{K,2j\}}$, respectively.
  - The $K-1$ depth of the tree has 2^K actions in total, referred to as $\alpha_{\{K,j\}}$ where $j \in \{1, ..., 2^K\}$.
  - The selected action denoted by: $\alpha_{\{K,j\}}$, is the child of $\mathcal{A}_{\{K-1,\lceil j/2 \rceil\}}$.
- **The actions at level K ($k = K$)**: At depth K, i.e., at leaves of the tree, we have the actions that directly interact with the Environment.
- $P_{\{k,j\}} = [p_{\{k+1,2j-1\}}, p_{\{k+1,2j\}}]$: The action probability vector of LA $\mathcal{A}_{\{k,j\}}$, where $k \in \{0, ..., K-1\}$ and $j \in \{1, ..., 2^k\}$.

Fig. 1 visualizes the structure of a simple HDPA when four actions exist in the Environment. The leaves of the tree are representations of the actions that the HDPA can take and interact with the Environment. For a HDPA with 2^K actions in the leaf level, the number of LA in the tree structure is $2^K - 1$. In this example, we have three LA, i.e., $\mathcal{A}_{\{0,1\}}$, $\mathcal{A}_{\{1,1\}}$ and $\mathcal{A}_{\{1,2\}}$. As depicted, each LA in the tree has two actions. To choose an action, HDPA follows the

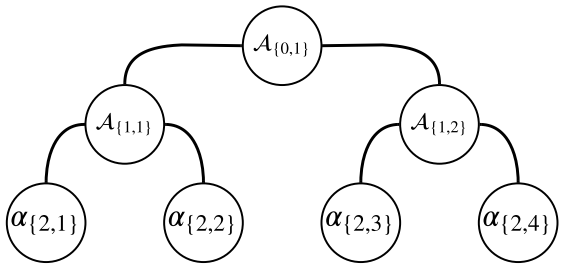

Fig. 1. A visualization of the hierarchy of the HDPA. The instances at the bottom/leaf level of the tree represent the actions that interact with the Environment.

path down the tree by sampling of these action probabilities in the vector. For example, when $\mathcal{A}_{\{0,1\}}$ has an action probability vector of [0.9, 0.1], it selects $\alpha_{\{1,1\}}$ at the root level with probability 0.9. Once $\alpha_{\{1,1\}}$ is chosen, $\mathcal{A}_{\{1,1\}}$ is selected for making the decision at level 1 for the next level (which is at the leaf level in this example), following the action probability vector that $\mathcal{A}_{\{1,1\}}$ maintains. Thereafter, if $\mathcal{A}_{\{1,1\}}$ happens to select the action $\alpha_{\{2,2\}}$, it means that the second action is chosen to interact with the Environment. Thereafter, the HDPA updates its parameters based on the feedback from the Environment. If and only if the Environment offers a Reward, following the pursuit concept, the action probabilities are updated, following the reverse path from the leaf with the current maximum reward estimate to the root. The reader should note that HDPA might reward another action than the one that is currently selected, due the inferior reward estimate in the currently selected action. Independent of whether a Reward or Penalty is received, the reward estimates in all the DPA instances are updated according to whether the action received a Reward or not. The process is detailed as follows:

Parameters:
Δ: The learning parameter, where $0 < \Delta < 1$, and its value is usually configured close to zero.
$u_{\{K,j\}}$: The number of times that action $\alpha_{\{K,j\}}$ was rewarded *when* selected, where $j \in \{1, \dots, 2^K\}$.
$v_{\{K,j\}}$: The number of times that action $\alpha_{\{K,j\}}$ was selected, where $j \in \{1, \dots, 2^K\}$.
$\hat{d}_{\{k,j\}}$: The estimated reward probability of action $\alpha_{\{k,j\}}$, $k \in \{1, ..., K\}$, $j \in \{1, \dots, 2^k\}$. At level K, $\hat{d}_{\{K,j\}}$ is computed as $\hat{d}_{\{K,j\}} = \frac{u_{\{K,j\}}}{v_{\{K,j\}}}$, where $j \in \{1, \dots, 2^K\}$.
β: The response from the Environment, where $\beta = 0$ corresponds to a Reward, and $\beta = 1$ to a Penalty.
T: Convergence criterion threshold.
We initialize the estimate of the reward probabilities as 0.5, i.e., $u_{\{K,j\}}(0) = 1$, $v_{\{K,j\}}(0) = 2$, thus $\hat{d}_{\{K,j\}}(0) = \frac{1}{2}$. The action probability vector is also initialized as 0.5 for all the LA, i.e., $P_{\{k,j\}}(0) = [\frac{1}{2}, \frac{1}{2}]$, where $k \in \{0, ..., K-1\}$ and $j \in \{1, ..., 2^k\}$.
Begin algorithm:
$t = 0$

```
Loop
  1. Depths 0 to $K-1$:
     – The LA $\mathcal{A}_{\{0,1\}}$ selects an action by randomly (uniformly) sampling as per its
       action probability vector $[p_{\{1,1\}}(t), p_{\{1,2\}}(t)]$.
     – Let $j_1(t)$ be the index of the chosen action at depth 0, 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 acti-
       vates the next LA at depth "2".
     – This process continues including depth $K-1$.
  2. Depth $K$:
     – Let $j_K(t)$ be the index of the chosen action at depth $K$ where $j_K(t) \in \{1, \ldots, 2^K\}$.
     – Update $\hat{d}_{\{K,j_K(t)\}}(t)$ based on the response from the Environment at the leaf
       depth, $K$:
           $u_{\{K,j_K(t)\}}(t+1) = u_{\{K,j_K(t)\}}(t) + (1 - \beta(t))$
           $v_{\{K,j_K(t)\}}(t+1) = v_{\{K,j_K(t)\}}(t) + 1$
           $\hat{d}_{\{K,j_K(t)\}}(t+1) = \frac{u_{\{K,j_K(t)\}}(t+1)}{v_{\{K,j_K(t)\}}(t+1)}$.
     – For all other "leaf actions", where $j \in \{1, ..., 2^K\}$ and $j \neq j_K(t)$:
           $u_{\{K,j\}}(t+1) = u_{\{K,j\}}(t)$
           $v_{\{K,j\}}(t+1) = v_{\{K,j\}}(t)$
           $\hat{d}_{\{K,j\}}(t+1) = \frac{u_{\{K,j\}}(t+1)}{v_{\{K,j\}}(t+1)}$.
  3. Define the reward estimate for all other actions along the path to the root, $k \in$
     $\{0, ..., K-1\}$ in a recursive manner, where the LA at any one level inherits the
     feedback from the LA at the level below:
         $\hat{d}_{\{k,j\}}(t) = \max\left(\hat{d}_{\{k+1,2j-1\}}(t), \hat{d}_{\{k+1,2j\}}(t)\right)$.
  4. Proceed to updating the action probability vectors in the LA *along the reverse
     path from the leaf with the current maximum reward estimate*, as follows:
     – By definition, each LA $j \in \{1, ..., 2^k\}$ at depth $k$, referred to as $\mathcal{A}_{\{k,j\}}$, where
       $k \in \{0, ..., K-1\}$, has two actions $\alpha_{\{k+1,2j-1\}}$ and $\alpha_{\{k+1,2j\}}$. Let $j^h_{k+1}(t) \in$
       $\{2j-1, 2j\}$ be the index of the larger element between $\hat{d}_{\{k+1,2j-1\}}(t)$ and
       $\hat{d}_{\{k+1,2j\}}(t)$.
     – Let $\overline{j^h_{k+1}(t)} = \{2j-1, 2j\} \setminus j^h_{k+1}(t)$ be the opposite action, i.e., the one that
       has the lower reward estimate.
     – Update $p_{\{k+1,j^h_{k+1}(t)\}}$ and $p_{\{k+1,\overline{j^h_{k+1}(t)}\}}$ using the estimates $\hat{d}_{\{k+1,2j-1\}}(t)$ and
       $\hat{d}_{\{k+1,2j\}}(t)$ as (for all $k \in \{0, ..., K-1\}$):
       If $\beta(t) = 0$ Then
           $p_{\{k+1,j^h_{k+1}(t)\}}(t+1) = \min\left(p_{\{k+1,j^h_{k+1}(t)\}}(t) + \Delta, 1\right)$,
           $p_{\{k+1,\overline{j^h_{k+1}(t)}\}}(t+1) = 1 - p_{\{k+1,j^h_{k+1}(t)\}}(t+1)$.
       Else
           $p_{\{k+1,\overline{j^h_{k+1}(t)}\}}(t+1) = p_{\{k+1,\overline{j^h_{k+1}(t)}\}}(t)$,
           $p_{\{k+1,j^h_{k+1}(t)\}}(t+1) = p_{\{k+1,j^h_{k+1}(t)\}}(t)$.
       EndIf
  5. For each $\mathcal{A}_{\{k,j\}}$, if either of its action probabilities $p_{\{k+1,2j-1\}}$ and $p_{\{k+1,2j\}}$ sur-
     passes a threshold $T$, where $T$ is a positive number that is close to unity, the action
     probabilities for the HDPA will stop updating, and *convergence* is achieved.
  6. $t = t + 1$
EndLoop
End algorithm
```

4 Experimental Results

To demonstrate the performance of the HDPA compared with the HCPA in [12], we simulated the different schemes' performance for distinct Environments. To enhance the validity of our simulations, we increased the number of experiments and the criteria for convergence compared with the simulations in [12]. As discussed previously, the HCPA has an impediment when the action probability vector approach unity. Our simulations, which we present in more detail shortly, demonstrate the advantage of the HDPA over the HCPA as the convergence criterion is high. The reader should note that we have omitted the comparison with the traditional CPA variants in this paper due to their well-known inferior performance [12].

We conducted experiments for different Environments with *many* actions. The Environments for the 16, 32, and 64 actions were based on the benchmark action probabilities that were first established in [12]. For the 128 actions Environment, we uniformly generated 128 different probabilities between zero and unity, representing the probabilities of the LA receiving a Reward from the Environment. The 128-action Environment's reward probabilities are visualized in Fig. 2.

4.1 The Learning Parameters

From the mathematical proof in [12], and the established theory of VSSA, we understand that when the learning parameter, λ, is sufficiently small, the HCPA will most likely converge to the action that ensures it the maximum probability of obtaining a Reward. The same applies to the value of Δ [14]. In general, a smaller learning parameter results in a slower convergence but has a higher probability of finding the optimal action as its configured value approaches zero. Therefore, tuning the learning parameter is a trade-off between the system's accuracy and convergence speed. In this paper, to find the best value of λ and Δ, we utilized a top-down approach. In more detail, we decreased the value of the learning parameters in a step-wise manner with two decimals precision until their configured value made the LA achieve 100% accuracy for all the given experiments. Consequently, the value of the learning parameters that fulfilled these criteria represents the assumed "best" values of λ and Δ given the distinct Environments used in the simulations.

Although the values for λ and Δ are obtained through extensive testing, it is not entirely certain that one will achieve convergence to the correct action with the found values of λ or Δ for a certain number of consecutive experiments. The Environment and the LA's stochastic behavior over time make it impossible to determine whether the LA will surely converge correctly with a λ or Δ below a certain threshold. The reader should also note that the λ and Δ values are entirely dependent on the Environment's reward probabilities and that the best λ and Δ can vary from case to case. Therefore, we refer to the obtained values of λ and Δ as the "best" learning parameters (and not the *optimal* ones).

4.2 The Average Number of Iterations

In the field of LA, a learning scheme's performance is often measured through the number of iterations required before the algorithm has converged. Due to the stochastic nature of the Environments that LA operates in, we normally measure the average number of iterations, i.e., we conduct many experiments and report the average number

Table 1. HCPA performance for the different simulation Environments.

Number of actions	Mean	Standard deviation (Std.)
16	1,366.61	121.14
32	10,281.84	681.82
64	169,839.67	13,687.48
128	155,088.62	10,613.21

Table 2. HDPA performance for the different simulation Environments.

Number of actions	Mean	Standard deviation (Std.)
16	868.25	135.50
32	6,172.38	744.84
64	100,638.41	17,653.41
128	97,795.59	13,266.12

of iterations required before convergence over a number of experiments. In VSSA, the LA has achieved convergence once the action probability of any one of the actions has reached a certain threshold. The convergence criterion threshold is often configured close to unity. In these simulations, we configured $T = 0.992$, and considered the average of the schemes' performance for 600 experiments.

The simulations discussed in this section, the "best" learning parameters for the Environment with 16 actions were $\lambda = 0.0043$ and $\Delta = 0.0011$. For the Environment with 32 and 64 actions, the best learning parameters were $\lambda = 0.00057$ and $\lambda = 3.6e{-5}$, and $\Delta = 0.00015$ and $\Delta = 9.9e{-6}$, respectively. The "best" obtained values for the 128 action Environment were $\lambda = 3.9e{-5}$ and $\Delta = 9.7e{-6}$. The reader will observe that the learning parameters for the 64 actions and 128 actions cases are quite similar, which is because the 64 actions' Environment's benchmark probabilities were more challenging than the generated Environment for 128 actions.

Tables 1 and 2 show results for our different simulation Environments. In these simulations, we used the benchmark probabilities for the Environments with 16, 32, and 64 actions. For the 128 actions Environment, we used the reward probabilities visualized in Fig. 2. The tables include both algorithms' results and list both the mean and the Standard Deviation (Std). Let us first consider the 16, 32, and 64 actions' Environments, where the HDPA had approximately 37%, 40%, and 41% fewer required iterations compared with the HCPA. Consequently, the benefit of HDPA over HCPA increased with the number of actions. Observing the Std, the HDPA had more variation in the number of iterations for all the three action configurations.

Considering the results obtained for the 128 actions' Environment (based on the reward probabilities visualized in Fig. 2), the HDPA converged within approximately $98,000$ iterations, while the HCPA required $155,000$ iterations before convergence. Comparing the algorithms' Std, the HDPA had more variation in the number of iterations before convergence than the HCPA. Thus, the HDPA was more unpredictable in its number of iterations required before convergence, but needed significantly fewer iterations on average!

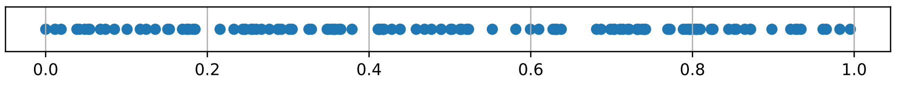

Fig. 2. The action probabilities for the 128 actions Environment.

4.3 The Nature of Convergence

The nature of convergence for the HDPA compared with the HCPA is different. As explained earlier, the HDPA updates its action probability in a discretized manner. In contrast, the HCPA updates these probabilities multiplicatively, where the increase in the probability vector can take any value in the interval $[0, 1)$. In Fig. 3 and Fig. 4, we can observe the differences between the operations of the HDPA and the HCPA in greater detail. In these experiments, we increased the convergence criterion to $T = 0.999$ and used the 64 actions Environment from the benchmark probabilities. The figures depict the different schemes' operation for a single iteration. We found the "best" learning parameters through the same top-down approach as explained earlier. The "best" learning parameters were configured as $\Delta = 1e{-}5$ and $\lambda = 5.3e{-}5$, respectively.

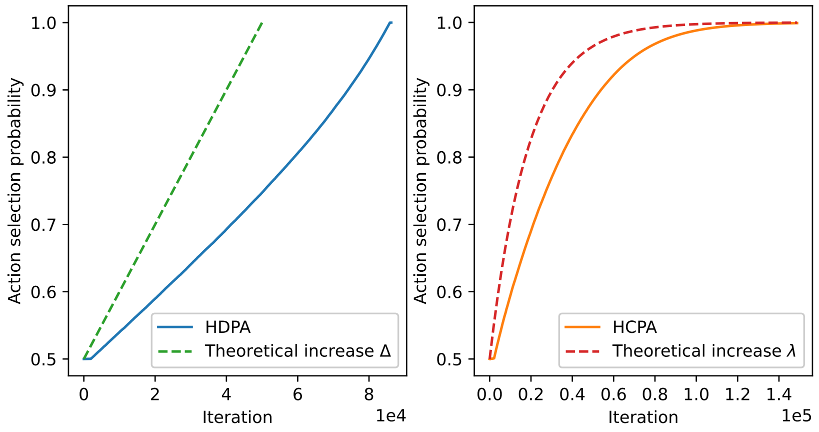

Fig. 3. The action probability of the optimal action per iteration compared to the theoretical increase in the action probability vector for the different schemes.

Fig. 3 depicts the action probability per iteration. As we can observe from the figure, the HDPA requires fewer iterations before convergence than the HCPA does. In addition, we observe the different nature of the two schemes. While the HDPA has a rather linear increase in the action probability, the increase of the action probability for the HCPA scheme slows down as the probability approach unity. Indeed, the HCPA has a faster increase than the HDPA in the initial iterations but suffers from slow convergence because of its behavior as the action probability increases. The reader

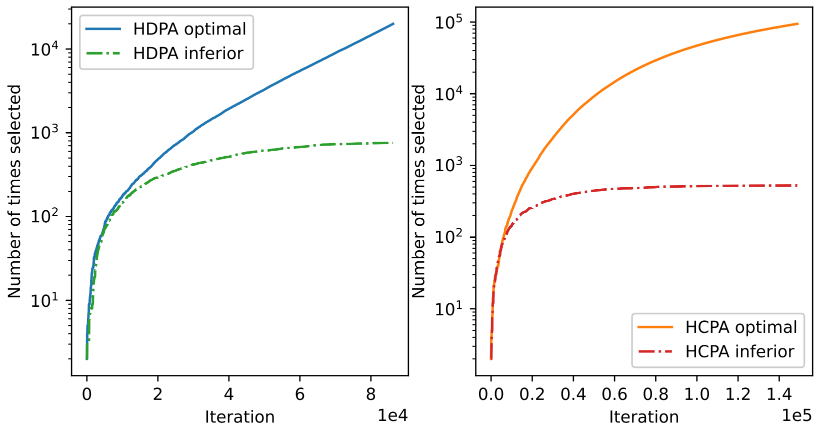

Fig. 4. The number of times that the optimal and the most inferior action were selected for the number of iterations required before convergence for the different schemes.

should also observe that both schemes do not follow the theoretical increase lines in the respective plots. The theoretical increase lines show the convergences when the optimal action is chosen in each iteration, where the action probabilities of the actions in the optimal branch are monotonically increasing. Clearly, these lines depict the theoretical convergence benchmarks without any exploration and are only achievable when the optimal action is known in advance.

In Fig. 4, we observe the difference between the optimal and inferior action for the different schemes. As we observe, the inferior action (the action with the lowest probability of resulting in a Reward) is only selected in the initial phase of the schemes' operation and rarely sampled as the algorithms approach convergence. In contrast, the optimal action is selected more often as the number of iterations increases. As we observe from the figure, the HDPA required significantly fewer iterations than the HCPA. The plots also demonstrate that the inferior action is, indeed, explored throughout the LA operation.

5 Conclusion

In this paper, we proposed the HDPA scheme. The HDPA incorporates all the major phenomena within LA that have improved these algorithms over the last six decades. By implementing VSSA probability updating functionality and discretizing the probability space, utilizing the Estimator phenomenon, and structuring the LA in a hierarchical tree structure akin to the concept of binary search, the HDPA outperforms the state-of-the-art HCPA when the convergence criterion is close to unity, i.e., when the accuracy requirement is high. Our simulations demonstrated the clear advantage of the HDPA over the HCPA for different Environments with many actions. We also demonstrated the difference between the action probability updating of the schemes, and thus, why their performance and convergence speeds are different.

References

1. Lakshmivarahan, S.: Learning Algorithms Theory and Applications. New York Springer-Verlag (1981)
2. Lakshmivarahan, S., Thathachar, M.A.L.: Absolutely expedient algorithms for stochastic automata. IEEE Transactions on Systems, Man, and Cybernetics **3**, 281–286 (1973)
3. Lanctot, J.K., Oommen, B.J.: Discretized estimator learning automata. IEEE Transactions on Systems, Man, and Cybernetics **22**(6), 1473–1483 (1992)
4. Narendra, K.S., Thathachar, M.A.L.: Learning Automata: An Introduction. Courier Corporation (Dec 2012)
5. Oommen, B.J.: Absorbing and ergodic discretized two-action learning automata. IEEE Transactions on Systems, Man, and Cybernetics **16**(2), 282–293 (1986)
6. Oommen, B.J., Christensen, J.P.R.: ϵ-optimal discretized linear reward-penalty learning automata. IEEE Transactions on Systems, Man, and Cybernetics **18**(3), 451–458 (1988)
7. 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)
8. Poznyak, A.S., Najim, K.: Learning Automata and Stochastic Optimization, vol. 3. Springer (1997)
9. Thathachar, M.A.L., Sastry, P.S.: Estimator algorithms for learning automata. Proceedings of the Platinum Jubilee Conference on Systems and Signal Processing (1986), Department of Electrical Engineering, Indian Institute of Science
10. Tsetlin, M.L.: Finite automata and the modeling of the simplest forms of behavior. Uspekhi Matem Nauk **8**, 1–26 (1963)
11. X. Zhang, B.J.O., Granmo, O.C.: The design of absorbing bayesian pursuit algorithms and the formal analyses of their ϵ-optimality. Pattern Analysis and Applications **20**, 797–808 (2017)
12. Yazidi, A., Zhang, X., Jiao, L., Oommen, B.J.: The hierarchical continuous pursuit learning automation: A novel scheme for environments with large numbers of actions. IEEE Transactions on Neural Networks and Learning Systems **31**(2), 512–526 (2020)
13. Zhang, X., Granmo, O.C., Oommen, B.J.: Discretized Bayesian pursuit - A new scheme for reinforcement learning. In: Proceedings of IEA-AIE. pp. 784–793. Dalian, China (Jun 2012)
14. Zhang, X., Jiao, L., Oommen, B.J., Granmo, O.C.: A conclusive analysis of the finite-time behavior of the discretized pursuit learning automaton. IEEE transactions on neural networks and learning systems **31**(1), 284–294 (2020)