The Hierarchical Discrete Pursuit Learning Automaton: A Novel Scheme With Fast Convergence and Epsilon-Optimality

Rebekka Olsson Omslandseter, Lei Jiao, *Senior Member, IEEE*, Xuan Zhang, Anis Yazidi, *Senior Member, IEEE*, and B. John Oommen, *Life Fellow, IEEE*

B. John Oommen dedicates this paper to Neil and Louise Lee, and Michael and Inger-Maria Twilley, who were like parents to his wife and him, when they moved to Canada in 1982.

***Abstract*— Since the early 1960s, the paradigm of learning automata (LA) has experienced abundant interest. Arguably, it has also served as the foundation for the phenomenon and field of reinforcement learning (RL). Over the decades, new concepts and fundamental *principles* have been introduced to increase the LA's speed and accuracy. These include using probability updating functions, discretizing the probability space, and using the "Pursuit" concept. Very recently, the concept of incorporating "structure" into the ordering of the LA's actions has improved both the speed and accuracy of the corresponding hierarchical machines, when the number of actions is *large*. This has led to the ϵ-optimal hierarchical continuous pursuit LA (HCPA). This article pioneers the inclusion of *all* the above-mentioned phenomena into a new single LA, leading to the novel hierarchical discretized pursuit LA (HDPA). Indeed, although the previously proposed HCPA is powerful, its speed has an impediment when any action probability is close to unity, because the updates of the components of the probability vector are correspondingly smaller when any action probability becomes closer to unity. We propose here, the novel HDPA, where we infuse the phenomenon of discretization into the action probability vector's updating functionality, and which is invoked recursively at every stage of the machine's hierarchical structure. This discretized functionality does not possess the same impediment, because discretization prohibits it. We demonstrate the HDPA's robustness and validity by formally proving the ϵ-optimality by utilizing the moderation property. We also invoke the submartingale characteristic at every level, to prove that the action probability of the optimal action converges to unity as time goes to infinity. Apart from the new machine being ϵ-optimal, the numerical results demonstrate that the number of iterations required for convergence is significantly reduced for the HDPA, when compared to the state-of-the-art HCPA scheme.**

***Index Terms*— Convergence analysis, hierarchical discrete pursuit LA, learning automata (LA), reinforcement learning (RL).**

I. Introduction

THE field of learning automata (LA), pioneered by Tsetlin [1] in the 1960s, has been thoroughly studied over the years [2]. In LA, nonhuman agents learn with the goal of solving particular tasks through computer programs. Specifically, the concept of LA is based on a *learning agent*, referred to as a LA,[1] interacting with a *teacher*, referred to as the Environment. LA entails lightweight adaptive learning schemes that are able to solve complex learning tasks in *stochastic* Environments. The LA learns from the Environment through continuous trial-and-error interactions, and gradually increases its chances of choosing the most favorable action. Without loss of generality, the mapping from the States to the Actions is deterministic.

The LA operates in conjunction with a stochastic Environment, where the LA chooses an action from among a finite set of *actions* offered by the Environment, which, in turn, provides a feedback based on the chosen action. Consequently, the LA adjusts its action based on a selection strategy as per this feedback. Hopefully, this feedback cycle should subsequently lead to the LA making "more intelligent" decisions. The feedback from the Environment is commonly binary, but it can also be from a finite set, or from a continuous range.

There are primarily two categories of LA, namely:

1) Fixed structure stochastic automata (FSSA), which have a fixed policy for the interstate transitions within a finite set of states, where the LA's current state corresponds to its chosen action [2], and where both the updating and decision functionalities are, typically, time-invariant.
2) Variable structure stochastic automata (VSSA), where the action selection is based on an action probability vector. In VSSA, updating *functions* are utilized to change the behavior of the LA according to feedbacks

Manuscript received 2 November 2021; revised 9 November 2022; accepted 29 November 2022. This work was supported by the European Economic Area (EEA) and Norway Grants 2014–2021 through the Project "Spacetime Vision: Toward Unsupervised Learning in the 4D World" under Grant EEA-RO-NO-2018-04. A preliminary and very abridged version of some of these results was presented at the 34th Australasian Joint Conference on Artificial Intelligence (AJCAI 2021), in February 2022, in Sydney, Australia. *(Corresponding author: B. John Oommen.)*

Rebekka Olsson Omslandseter and Lei Jiao are with the Department of Information and Communication Technology, University of Agder, 4879 Grimstad, Norway (e-mail: rebekka.o.omslandseter@uia.no; lei.jiao@uia.no).

Xuan Zhang is with the Norwegian Research Center (NORCE), 4879 Grimstad, Norway (e-mail: xuan.z.jiao@gmail.com).

Anis Yazidi is with the Department of Computer Science, Oslo Metropolitan University, 0160 Oslo, Norway (e-mail: anisy@oslomet.no).

B. John Oommen is with the School of Computer Science, Carleton University, Ottawa, ON K1S 5B6, Canada, also with the University of Agder, 4879 Grimstad, Norway, and also with the TRADE Research Entity, North-West University, Potchefstroom 2520, South Africa (e-mail: oommen@scs.carleton.ca).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3226538.

Digital Object Identifier 10.1109/TNNLS.2022.3226538

[1]In this article, the shortened term LA refers to both the field of LA, the machine, and the Learning Automaton itself, depending on the context in which it appears.

from the Environment. These will be explained in more detail in Section II.

LA have scores of applications reported in the Literature. In the interest of brevity, we omit[2] them here and merely include them in the bibliography.

A. Goal of This Article

In any competition, setting an initial record is hard enough,[3] but excelling it is a feat. The goal of this article is to try to attain to a speed/accuracy limit for LA dealing with a large number of actions, that will be hard (if not impossible) to beat!

Improving the speed and accuracy of LA has always involved discovering new concepts and fundamental *principles*. One of the aims of this article is to record the ways by which the speed/accuracy of LA has been improved over the last six decades. All of these enhancements have incorporated new and fundamental principles that were not invented earlier. Subsequently, combinations of these principles have led to even further improvements. Our goal is to record all of the principles and paradigms, and then combine them efficiently.[4]

As briefly explained below, these include using probability updating functions in VSSA, discretizing the probability space, and using the "Pursuit" concept. Very recently, the concept of incorporating "structure" into the ordering of the LA's actions has improved both the speed and accuracy of the corresponding *hierarchical* machines, when the number of actions is *large*. This has led to the ϵ-optimal hierarchical continuous pursuit LA (HCPA), which is currently the record holder for such Environments. This article pioneers the inclusion of *all* the above-mentioned phenomena into a new single LA, leading to the novel hierarchical discretized pursuit LA (HDPA).

B. Organization of This Article

The remainder of the article is organized as follows. Section II takes the reader through all the avenues by which the speed/accuracy of LA has been enhanced in quantum jumps or relatively incrementally, over the last six decades. This motivates and sets the stage for the main contribution of this article, namely the HDPA. In Section III, we describe, in detail, the new algorithm. In Section IV, we prove the algorithm's convergence property, i.e., its ϵ-optimality. The numerical results are presented in Section V, after which we conclude the article in Section VI.

II. Strategies to Enhance Speed/Accuracy in LA

A. Infancy: FSSA

As briefly alluded to above, in a stochastic LA, if the state transition probability and output function are constant, i.e., they do not vary with the time step "t" and the input sequence, the automaton is an FSSA. The pioneering and popular examples of these LA were proposed by Tsetlin [1], Krylov, and Krinsky—all of which are ϵ-optimal under various conditions. These were the primitive ground-breaking LA, and the whole world of LA and reinforcement learning (RL) had their very existence because of them. Their details can be found in [2].

B. From FSSA to VSSA

The first quantum increase in speed was achieved by the discovery/invention of VSSA. Unlike FSSA, VSSA are the ones in which the state transition probabilities are not fixed. Here, the state transitions or the action probabilities themselves are updated at every time instant using a suitable scheme.

VSSA are an *order of magnitude* faster than FSSA because:

1) VSSA permit an enhanced stochastic exploration of the action probability space, rather than moving through the states of the machine step-by-step, as FSSA do;
2) unlike FSSA, VSSA permit a "switch" of actions at every step in time, and not merely at the so-called Boundary states;
3) VSSA also provide a far greater flexibility, because they utilize functions to determine the updating, and the number of functions that can be used is limitless;
4) the transition probabilities and the output function vary with time, and the action probabilities are updated on the basis of the input. The action chosen is dependent on the action probability vector, which is, in turn, updated based on the Reward/Penalty input that the LA receives from the Environment.

VSSA are modeled by a discrete-time Markov Process, defined on a suitable set of states. If a probability updating scheme is time-invariant, the action probability vector when $t \geq 0$, $\{P(t)\}_{t\geq 0}$, is a discrete-time, homogenous Markov process, and the probability vector at the current time instant $P(t)$, [along with the action at time t, $\alpha(t)$, and the feedback from the Environment at time t, $\beta(t)$] completely determine $P(t+1)$. Hence, each distinct updating scheme identifies a different type of learning algorithm. For *Continuous Linear* VSSA, the following four learning schemes are extensively studied in the literature: The well-known Linear Reward-Penalty ($L_{\text{R–P}}$) scheme, the Linear Reward-Inaction ($L_{\text{R–I}}$) scheme, the Linear Inaction-Penalty ($L_{\text{I–P}}$) scheme, and the Linear Reward-ϵPenalty ($L_{\text{R}-\epsilon\text{P}}$) scheme are examples of *linear* VSSA updating rules [2], [3]. As opposed to these, increasing the probabilities of the LA in a nonlinear manner has been investigated in [2], [3], and [23].

C. From Continuous VSSA to Discretized VSSA

The next paradigm that was invented/discovered to increase the speed/accuracy of LA was that of discretizing the action probability space. The previous VSSA algorithms are *continuous*, i.e., the action probabilities can assume any real value in the interval $[0, 1]$. In such LA, the choice of an action is determined by a random number generator (RNG). In order to increase the speed of convergence of these LA, Oommen [24] introduced the family of *discretized* algorithms which pioneered the discretization of the probability space.

Discretized automata can be perceived to be like a hybrid combination of FSSA and VSSA. Discretization is conceptualized by restricting the probability of choosing the actions to only a fixed number of values in the closed interval $[0, 1]$.

[2]The original submission had a detailed list of the applications of LA from the past decades. Since they are all well cited in the Literature, we merely include them in the bibliography as per the request of the AE and Referees. We are grateful for their input. They can be included if required by the EiC.

[3]This is apparent from the 100-m sprint, which is reckoned as the ultimate test of a person's speed. The physical and psychological barrier of completing it in under 10 s makes the person "a world-class sprinter." Although Carl Lewis pioneered this challenge at 9.97 s in 1983, the current record holder is Usain Bolt who ran it in 9.58. It takes a lot more effort, and years of hard work, to even marginally improve a quantifying performance metric, that is almost at the limit of *par excellence*!

[4]We are not aware of any publication which records these details, and we believe that it will be extremely helpful for future researchers.

Thus, the updating of the action probabilities is achieved in steps, rather than in a continuous manner. The different properties (absorbing and ergodic) of these LA, and the updating schemes of action probabilities for these discretized automata (like their continuous counterparts) were later studied in detail by Oommen [24] and Oommen and Christensen [25]. Also, similar to the continuous LA paradigm, the discretized versions, the DL$_{\text{RI}}$, DL$_{\text{IP}}$, and DL$_{\text{RP}}$ automata have also been reported.

Originally, the assumption was that the RNGs could generate real values with arbitrary precision. In the case of discretized LA, if an action probability is reasonably close to unity, the probability of choosing that action increases *in a single iteration* to unity (when the conditions are appropriate) directly, rather than asymptotically.

The second important advantage of discretization is that it is more practical since RNGs used by continuous VSSA can only *theoretically* be assumed to be *any* value in the interval [0, 1]. But machine implementations use *pseudo*-RNGs, where the set of possible values is not infinite in [0, 1], but finite.

Finally, discretization is also important in terms of implementation and representation. Discretized implementations of automata use *integers* for tracking the number of multiples of the learning parameter, $(1/N) = \Delta$, where N is the so-called *resolution* parameter. This, not only increases the rate of convergence of the algorithm, but also reduces the time, in terms of the clock cycles it takes for the processor to do each iteration of the task, and the memory needed. By virtue of the above, discretized algorithms are both more time and space efficient than their continuous counterpart algorithms.

D. Estimator-Based Paradigm

The next major quantum jump in the speed/accuracy of LA was by the discovery/invention of estimator-based algorithms (EAs). Just as in the case of the family of discretized algorithms, Thathachar and Sastry designed a new class of algorithms, called the *Estimator Algorithms* [4], which at their time possessed a faster rate of convergence than all the previous families. These algorithms, like the previous ones, maintain and update an action probability vector. However, unlike the previous ones, these algorithms also keep running estimates for each action that is rewarded, using a *reward-estimate vector*, and then use those estimates in the probability updating equations. The reward estimates vector is, typically, denoted in the literature by $\hat{D}(t) = [\hat{d}_1(t), \ldots, \hat{d}_r(t)]^T$. The corresponding state vector is denoted by $Q(t) = \langle P(t), \hat{D}(t)\rangle$, and the estimates can be computed using a Maximum Likelihood scheme (see below), or in a Bayesian manner [26].

The reason for the quantum increase in speed is because in EAs, the convergence involves two intertwined phenomena, namely the convergence of the reward estimates, $\hat{D}(t)$, and the convergence of the action probabilities themselves. The combination of these vectors in the updating rule is intricate, and must be done in a delicately designed manner. By the law of large numbers, if the actions are sampled "enough number of times," their estimates[5] converge to their true values. The Environment thus influences the probability vector both directly and indirectly, the latter being as a result of the estimation of the reward estimates of the different actions. This may, thus, lead to increases in action probabilities for actions different from the currently rewarded action. This revolutionary concept changed the entire world of LA, and indeed, even though there is an added computational cost involved in maintaining the reward estimates, these estimator algorithms possess an order of magnitude superior performance than the nonestimator algorithms previously introduced.

Pursuit algorithms are a subclass of EAs that *pursue* an action that the automaton "currently" perceives to be optimal. The first pursuit algorithm, referred to as the CP_{RP} algorithm due to Thathachar and Sastry, pursues the optimal action on Reward and Penalty. Here, the *currently perceived* "best action" is rewarded, and *its* action probability value is increased with a value directly proportional to its distance to unity, whereas the "less optimal actions" are penalized. The cases of changing the action probabilities *only* on reward and ignoring the penalties lead to the CP$_{\text{RI}}$ scheme, also described in [27]. Thathachar and Sastry [4] introduced the class of continuous EAs, where one pursues not only the best currently optimal action,[6] and Agache and Oommen [27] proposed the so-called Generalized Pursuit LA.

E. Merging Estimators-Based and Discretized Worlds

The next steps in enhancing the speed/accuracy of LA involved merging the properties of the previously introduced phenomena. In particular, the researchers piggy-backed on the benefits of discretization and of the Pursuit paradigm. Utilizing the previously proven capabilities of discretization in improving the speed of convergence of the learning algorithms, Lanctot and Oommen [28] enhanced the Pursuit algorithm and the "Thathachar and Sastry's Estimator" algorithm [4]. This led to the designing of classes of learning algorithms, referred to in the literature as the discrete estimator algorithms (DEAs) [28], which possessed the so-called *Moderation* and *Monotone* Properties. Agache and Oommen [27] provided a discretized version of their GPA algorithm presented earlier. Their algorithm, called the Discretized Generalized Pursuit Algorithm (DGPA), also essentially generalized the "Thathachar and Sastry's Estimator" algorithm [4].

All of these were further investigated in [29] and [30], respectively, where the earlier flawed proofs of the schemes themselves were perfected.

F. Incorporating Structure

All of the LA schemes that have been discussed till now assumed that the actions were unordered, which, of course, makes sense since the penalty probabilities are unknown. Indeed, why should one action be preferred above the others? The next major quantum jump in the speed and accuracy of designing LA occurred by incorporating *structure* into the ordering of the actions. This represents the current state-of-the-art, and is particularly pertinent when the number of actions involved, R, is large. In such scenarios, the learning problem becomes extremely complex, which motivated

[5]The convergence proofs of EAs are far more complex than those of traditional LA. This is because, if the accuracies of the estimates are poor because of inadequate estimation (i.e., if the suboptimal actions are not sampled "enough number of times"), the convergence accuracy can be diminished. We address this issue later.

[6]The probability updates differ depending on whether the reward estimates are smaller or larger than the estimate of the currently selected action.

Yazidi et al. [31] to devise a scheme by which small subsets of actions (e.g., of cardinality two) were compared, and the result of their comparison was trickled up to avoid dealing with R-action LA and vectors.

Unlike the prior art, in the case of FSSA, one requires S-states for each of the R actions. When the number of actions is large, an LA deals with an $R \cdot S \times R \cdot S$-sized Markov chain, and this adds to the sluggishness of the machine. In the case of VSSA, the action probability vector has a dimension of R 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. For the families of pursuit algorithms, the problem still exists because one still utilizes the action probability vector with dimension R, which could be large in this setting.

To make the LA work for a large number of actions, the HCPA was developed [31]. In the hierarchical structure[7] of the HCPA, instead of using one CPA with R actions, they employed multiple CPA, and arranged them in different layers. In this way, the authors avoided having insignificant values in the action probability vector. This endowed the HCPA with the ability to handle the cases when R was very large, which was not even feasible for the traditional CPA to solve.

The quantum jump in speed and accuracy was achieved by merging the phenomena of VSSA and EAs, and doing this in an ordered hierarchical manner, where each LA dealt with a small number of actions. Indeed, the convergence speed of the novel LA proposed in [31] was *many* orders of magnitude faster than any of the other legacy LA. In essence:

1) It incorporated the area of "data structures" into the field of LA, and suggested a novel *hierarchical* LA which uses a tree structure as a part of the learning process;
2) the scheme was based on a multilevel hierarchy composed of two-action CPA at each of the levels, where both the estimation required for an EA, and interaction occurred only at the leaves of the hierarchy;
3) the individual LA performed the learning locally and the result of this was trickled-up in a recursive manner by considering *only* a node and its sibling so as to achieve *global learning*. This also mitigated the problem of having very small action probabilities, since every LA dealt with only two actions.

The HCPA is the state-of-the-art in LA. The goal of this article is to incorporate all of the above phenomena (VSSA, discretization, the Estimator phenomenon and structure) into our present novel contribution, namely the HDPA. The contributions of this present work are thus summarized as follows.

1) We propose a novel HDPA that converges faster than the state-of-the-art HCPA algorithms as the convergence criterion is configured close to unity, e.g., above 0.99. The advantage of the HDPA over the HCPA becomes more obvious when one works with a *large* number of actions.
2) We prove, using a formal, rigorous mathematics analysis, the ϵ-optimality of HDPA.
3) By resorting to simulation results, we quantify, in detail, how much faster the convergence of the HDPA is when compared to the HCPA. We have also stated, for the first time, a bound for the number of iterations, which is an avenue for future analytical studies.

[7]A notable prior attempt to devise hierarchical LA is due to Papadimitriou [32]. The difference between what the HCPA and we have done (when compared to the work of [32]) is explained, in detail, in [31].

G. Roadmap for HDPA

Although the HCPA can work in the scenario when there are a large number of actions, the novelty of this article is that we have proposed a viable mechanism by which *its* convergence speed can be improved. By some insight, one observes that HCPA has a relatively sluggish convergence, especially when the required convergence accuracy is high, e.g., above 0.99. The reason behind it is that the changes in the probability vector decrease with the number of iterations. As the learning continues, the increment of the superior action probability is correspondingly decreased, making it more difficult to converge in the later phase of learning. To overcome this, we propose the HDPA to speed up the convergence when high convergence accuracy is required. The beauty of HDPA is that learning speed is not decreasing as the learning continues. This is because we piggy-back the phenomenon of discretization—we incorporate all of the above phenomena, i.e., VSSA, discretization, the Estimator phenomenon and structure!

The newly proposed discretized learning is shown to be faster than what has been achieved previously in the literature for Environments where the convergence criterion is configured to be more than 99%. The speed of the scheme for such convergence criteria is significantly faster than the HCPA, and the gained efficiency is observed to increase as the number of actions increases. Thus, when we are faced with many actions and we are concerned with the accuracy of convergence, the HDPA outperforms the state-of-the-art HCPA scheme presented in [31] in terms of efficiency, i.e., the number of iterations required before convergence.

We conjecture that the HDPA has thus attained to a speed/accuracy limit for LA dealing with a large number of actions, that will be hard (if not impossible) to beat!

III. Description of the HDPA

The HDPA incorporates all of the phenomena detailed in the introduction. More specifically, we organize a set of DPA instances in a hierarchical tree structure, where all the DPA instances have a set of actions corresponding to the possible paths down the tree structure. We maintain the action probabilities of the respective LAs through vectors that are updated in a discretized manner based on whether an action receives a Reward (or Penalty). At the bottom level of the tree, we have the actions that directly interact with the Environment, and we maintain reward estimates of all the different actions throughout the tree. According to the Pursuit concept, the HDPA pursues the currently estimated best action in all iterations. Thus, the action that currently has the best-estimated reward probability is rewarded upon a Reward, regardless of which action actually triggered the Reward. A more throughout explanation is given below.

A. Structure of the HDPA

In the interest of simplicity and clarity, in the following explanations, we will utilize two-action DP_{RI} instances as the primitive machine in the construction of the HDPA. Consequently, the hierarchy can be organized as a balanced

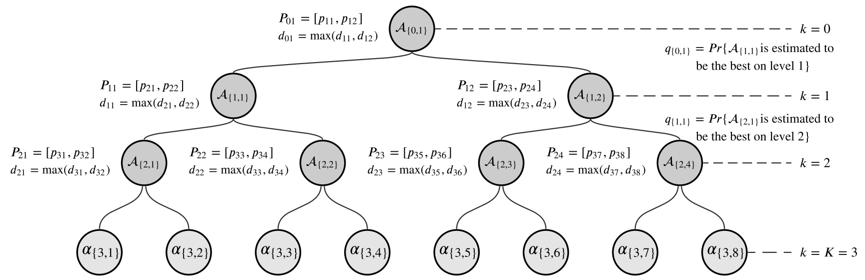

$P_{01} = [p_{11}, p_{12}]$
$d_{01} = \max(d_{11}, d_{12})$
$\mathcal{A}_{\{0,1\}}$
$k = 0$
$q_{\{0,1\}} = Pr\{\mathcal{A}_{\{1,1\}}$ is estimated to be the best on level 1$\}$
$P_{11} = [p_{21}, p_{22}]$
$d_{11} = \max(d_{21}, d_{22})$
$\mathcal{A}_{\{1,1\}}$
$P_{12} = [p_{23}, p_{24}]$
$d_{12} = \max(d_{23}, d_{24})$
$\mathcal{A}_{\{1,2\}}$
$k = 1$
$q_{\{1,1\}} = Pr\{\mathcal{A}_{\{2,1\}}$ is estimated to be the best on level 2$\}$
$P_{21} = [p_{31}, p_{32}]$
$d_{21} = \max(d_{31}, d_{32})$
$\mathcal{A}_{\{2,1\}}$
$P_{22} = [p_{33}, p_{34}]$
$d_{22} = \max(d_{33}, d_{34})$
$\mathcal{A}_{\{2,2\}}$
$P_{23} = [p_{35}, p_{36}]$
$d_{23} = \max(d_{35}, d_{36})$
$\mathcal{A}_{\{2,3\}}$
$P_{24} = [p_{37}, p_{38}]$
$d_{24} = \max(d_{37}, d_{38})$
$\mathcal{A}_{\{2,4\}}$
$k = 2$
$\alpha_{\{3,1\}}$
$\alpha_{\{3,2\}}$
$\alpha_{\{3,3\}}$
$\alpha_{\{3,4\}}$
$\alpha_{\{3,5\}}$
$\alpha_{\{3,6\}}$
$\alpha_{\{3,7\}}$
$\alpha_{\{3,8\}}$
$k = K = 3$

Fig. 1. Visualization of the tree structure in the HDPA with notations as utilized for the explanations in the article.

full binary tree for a problem with 2^K actions[8] and a maximum depth of K. If the number of actions is less than 2^K, one can add dummy actions with zero reward probabilities. Likewise, when we have more than 2^K actions, we need to consider the nearest power of two and then set the action probabilities to zero for the excess number of actions. To incorporate the mathematics established in [31] and for ease of the comparisons to the HCPA scheme, we utilize notations that are similar to those of the latter paper. We further formalize the levels in the hierarchy as follows.

1) *The Depth Index of the Tree:* For a tree with the maximum depth K, we employ k to index the depth of the tree, where $k \in \{0, 1, \ldots, K\}$.
2) *The Various LA:* We denote a specific LA by $\mathcal{A}_{\{k,j\}}$, where k refers to its depth and j represents its particular index in depth, k. More specifically, the LA $j \in \{1, \ldots, 2^k\}$ at depth k is referred to as $\mathcal{A}_{\{k,j\}}$, where $k \in \{0, \ldots, K-1\}$. The LA at the top of the hierarchy is the one at depth 0.
3) *The LA at Depths From* 0 *to* $K-1$ *(*$0 \leq k < K-1$*):*
   a) each of the LA, $\mathcal{A}_{\{k,j\}}$, has two actions, denoted by $\alpha_{\{k+1,2j-1\}}$ and $\alpha_{\{k+1,2j\}}$, respectively;
   b) whenever the action $\alpha_{\{k+1,2j-1\}}$ is chosen, the specific LA $\mathcal{A}_{\{k+1,2j-1\}}$ at the next level is activated;
   c) whenever the action $\alpha_{\{k+1,2j\}}$ is chosen, the specific LA $\mathcal{A}_{\{k+1,2j\}}$ at the next level is activated;
   d) $\mathcal{A}_{\{k+1,2j-1\}}$ and $\mathcal{A}_{\{k+1,2j\}}$ are referred to as the *Left Child* and the *Right Child* of its parent ($\mathcal{A}_{\{k,j\}}$), respectively.
4) *The LA at Depth* $K-1$ *(*$k = K-1$*):* The LA at depth $K-1$ select the actual actions to interact with the Environment:
   a) 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;
   b) the $K-1$ depth has 2^K actions in total, referred to as $\alpha_{\{K,j\}}$ where $j \in \{1, \ldots, 2^K\}$;
   c) the selected action denoted by: $\alpha_{\{K,j\}}$ is the child of $\mathcal{A}_{\{K-1,\lceil j/2 \rceil\}}$.
5) *The Actions at Level K (*$k = K$*):* At depth K, i.e., at the bottom of the tree, we have the actions that directly interact with the Environment.

[8]The LA instances can easily be extended to have more actions, but the reader should remember that we endeavor to mitigate the problem of slow convergence associated with *many* actions in the action probability vector. Consequently, the number of actions should, in any case, be limited in consideration of the convergence rate.

B. Concept of the HDPA

As explained above, the concept of the HDPA is to organize the DPA nodes in a tree structure. Observe that any of the various reported instantiations of the DPA can be utilized at every level. However, we have chosen to utilize the DP_{RI} instances, because the Reward-Inaction scheme has demonstrated a superior performance than the Reward-Penalty types [31].

As depicted in Fig. 1, each node, except the nodes at the bottom level, is the parent of two children, i.e., each node maintains a discretized probability vector with two possible actions, corresponding to its children. The HDPA maintains the original actions through the two-actions DPA instances at the second bottom level of the tree, i.e., the nodes at depth $K-1$. Consequently, if a problem has $2^8 = 256$ actions, there are 128 nodes at the $K-1$ depth of the tree, each maintaining a two-action probability vector, i.e., 256 actions in total.

By way of example, let us consider the structure in Fig. 1, where we have eight original actions, i.e., eight leaves. In this case, we have seven LA, i.e., $\mathcal{A}_{\{0,1\}}$, $\mathcal{A}_{\{1,1\}}$, $\mathcal{A}_{\{1,2\}}$, $\mathcal{A}_{\{2,1\}}$, $\mathcal{A}_{\{2,2\}}$, $\mathcal{A}_{\{2,3\}}$, and $\mathcal{A}_{\{2,4\}}$. When the HDPA scenario can be structured as a full binary tree, the number of LA needed is given by $2^K - 1$. Each LA maintains an action probability vector of dimension 2. To choose an action, we follow the path down the tree by sampling the action probabilities in these vectors. 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, with probability 0.9. Once $\alpha_{\{1,1\}}$ is chosen, $\mathcal{A}_{\{1,1\}}$ is selected for making the decision at depth 1 for the next depth in the tree. Let us assume that $\mathcal{A}_{\{1,1\}}$ happens to select the action $\alpha_{\{2,2\}}$, and the LA $\mathcal{A}_{\{2,2\}}$ is consequently activated. After that, $\mathcal{A}_{\{2,2\}}$ selects the action at the leaf depth as per its action probability vector. If $\alpha_{\{3,4\}}$ happens to be chosen, the fourth original action is selected to interact with the Environment in that iteration. The HDPA consequently updates the probability components in the tree based on the feedback from the Environment. More specifically, the reward estimates are updated in the reverse path of the actions selected in the tree. For the updating of the action selection probabilities, it is based on whether the selected leaf receives a Reward or not. If the selected leaf receives a Reward, following the Pursuit concept, we reward all actions in the tree along the path that leads to the leaf with the current best reward estimate. Note that the leaf with the current best reward estimate may not be the selected leaf in the current iteration due to the fact that we only pursue the action that currently most likely receives a Reward, when averaged over all iterations. The above concepts are formalized below.

Notations and Definitions: To clarify the concepts explained above, we use the following definitions in the description of Algorithm 1.

1) 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 only two actions that can be selected at depth $K-1$, namely $\mathcal{A}_{\{K-1,j\}}$.
2) Each LA $j \in \{1, \ldots, 2^k\}$ at depth k, referred to as $\mathcal{A}_{\{k,j\}}$, where $k \in \{0, \ldots, K-1\}$ has two actions, namely $\alpha_{\{k+1,2j-1\}}$ and $\alpha_{\{k+1,2j\}}$.
3) $P_{\{k,j\}} = [p_{\{k+1,2j-1\}}, p_{\{k+1,2j\}}]$, is the action probability vector of LA $\mathcal{A}_{\{k,j\}}$, where $k \in \{0, \ldots, K-1\}$ and $j \in \{1, \ldots, 2^k\}$.

Parameters:

1) Δ*:* The learning parameter, where $0 < \Delta < 1$, and its value is configured arbitrarily close to zero.
2) $u_{\{K,j\}}$*:* The number of times that action $\alpha_{\{K,j\}}$ was rewarded *when* selected, where $j \in \{1, \ldots, 2^K\}$.
3) $v_{\{K,j\}}$*:* The number of times that action $\alpha_{\{K,j\}}$ was selected, where $j \in \{1, \ldots, 2^K\}$.
4) $\hat{d}_{\{k,j\}}$*:* The estimated reward probability of action $\alpha_{\{k,j\}}$, $k \in \{1, \ldots, K\}$, $j \in \{1, \ldots, 2^k\}$. At level K, $\hat{d}_{\{K,j\}}$ is computed as $\hat{d}_{\{K,j\}} = (u_{\{K,j\}}/v_{\{K,j\}})$, where $j \in \{1, \ldots, 2^K\}$.
5) β*:* The response from the Environment, where $\beta = 0$, corresponds to a Reward, and $\beta = 1$ to a Penalty.
6) T*:* The convergence criterion threshold.

We initialize the estimates of the reward probabilities as 0.5. Thus, both actions in all the LA in the tree have an initial estimated reward probability of 0.5, i.e., $u_{\{K,j\}}(0) = 1$, $v_{\{K,j\}}(0) = 2$, thus $\hat{d}_{\{K,j\}}(0) = (1/2)$. The action probability vector is also initialized as 0.5 for all the LA, i.e., $P_{\{k,j\}}(0) = [(1/2), (1/2)]$, where $k \in \{0, \ldots, K-1\}$ and $j \in \{1, \ldots, 2^k\}$.

Algorithm 1 can be simplified because it is unnecessary to update the reward estimates along the path for the algorithm to run. In other words, we only need the estimated reward probabilities for the actions that directly interact with the Environment, i.e., for the leaves. But we have chosen to describe the scheme using Algorithm 1, because we utilize, in Section IV, the reward estimates along the path in the convergence analysis. In the interest of space and brevity, as one can see, the detailed descriptions of the *updates* are omitted here, as recommended by the Referees. The interested reader can find them in [40].

IV. Proof of ϵ-Optimality

The proof follows the four-step method established in [33] for the DPA. But in contrast, we prove here that convergence will also occur when the learning units are structured hierarchically. In particular, we consider the moderation property and prove that we have a marginality property along the optimal path. After that, we utilize the submartingale property in a level-by-level approach. We finally prove that the probability of the optimal action approaches unity as time goes to infinity.

A. Moderation Property

We first need to consider the moderation property, proving that under the HDPA, by utilizing a sufficiently small value of the learning parameter, Δ, each action will be selected an arbitrarily large number of times.

Algorithm 1 The HDPA

```
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 activated LA, i.e., $\mathcal{A}_{\{1,j_1(t)\}}$, in turn, chooses an action and activates 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 action chosen 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 $j \in \{1, \ldots, 2^K\}$ 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 estimates for all other actions along the path to the root, $k \in \{0, \ldots, K-1\}$ in a recursive manner, where the LA at any one level inherits the feedback from the LA at the level below as
         $\hat{d}_{\{k,j\}}(t) = \max(\hat{d}_{\{k+1,2j-1\}}(t), \hat{d}_{\{k+1,2j\}}(t)).$
  4) Proceed to update the action probability vectors along the path leading to the leaf with the current maximum reward estimate, as follows.
     • By definition, each LA $j \in \{1, \ldots, 2^k\}$ at depth $k$, referred to as $\mathcal{A}_{\{k,j\}}$, where $k \in \{0, \ldots, K-1\}$ 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 with the lower reward estimate.
     • For all $k \in \{0, \ldots, K-1\}$, 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:
       If $\beta(t) = 0$ Then
         $p_{\{k+1,j^h_{k+1}(t)\}}(t+1) =$
                 $\min(p_{\{k+1,j^h_{k+1}(t)\}}(t) + \Delta, 1),$
         $p_{\{k+1,\overline{j^h_{k+1}(t)}\}}(t+1) = 1 - p_{\{k+1,j^h_{k+1}(t)\}}(t+1).$
       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 $p_{\{k+1,2j\}}$ surpasses 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

Theorem 1: For any value of $\delta \in (0, 1]$ and integer $M < \infty$, there exist a nonzero positive learning parameter, $\Delta_0 < 1$, such that for all $\Delta < \Delta_0$

$$\Pr\{x\} > 1 - \delta$$

where x indicates the event that each action is selected more than M times before time t_0.

Proof: The details of the proof that Theorem 1 is true for the DPA can be found in [4], [29] and [34]. We now further elaborate on why this is also true for the HDPA.

In the case of the HDPA, we have 2^K actions at depth K denoted as $\alpha_{\{K,j\}}$ and $j \in \{1, \ldots, 2^K\}$. Let $Y^t_{\{K,j\}}$ be the number of times action $\alpha_{\{K,j\}}$ is chosen up to time t. To prove Theorem 1, we want to show $\Pr\{Y^t_{\{K,j\}} > M\} > 1-\delta$, which is the same as $\Pr\{Y^t_{\{K,j\}} \leq M\} \leq \delta$. The events $\{Y^t_{\{K,j\}} = l\}$ and $\{Y^t_{\{K,j\}} = n\}$ are mutually exclusive for $l \neq n$. Consequently, it follows that:

$$\Pr\left\{Y^t_{\{K,j\}} \leq M\right\} = \sum_{l=1}^{M} \Pr\left\{Y^t_{\{K,j\}} = l\right\}.$$

Further, the probability of the actions at depth K being chosen is connected to the probabilities in shallower depths as follows:

$$\Pr\{\alpha_{\{K,j_K\}} \text{is chosen}\} = p_{\{K,j_K\}} p_{\{K-1,j_{K-1}\}}, \ldots, p_{\{0,j_0\}}$$

where $j_{K-1} = \lceil j_K/2 \rceil$, $j_{K-2} = \lceil j_{K-1}/2 \rceil, \ldots,$ and $j_0 = \lceil j_1/2 \rceil$. Considering the time aspect, it follows that:

$$\Pr\{\alpha_{\{K,j_K\}} \text{is chosen at time } t\} = p_{\{K,j_K\}}(t) p_{\{K-1,j_{K-1}\}}(t), \ldots, p_{\{1,j_1\}}(t).$$

Let us further assume that all the LA instances have the same action probability at beginning, i.e., $p_{k,j}(0) = 1/2$ at $t = 0$. For ease of expression, we use $p(0)$ to represent the initial action probabilities, $p_{k,j}(0)$, for all actions. By the *modus operandus* of the various instances of the DPA, we know that the magnitude by which any action probability can decrease in any single iteration is bounded by Δ such that we have

$$\Pr\{\alpha_{\{K,j_K\}} \text{ is chosen at time } t\} \leq 1 \text{ and}$$
$$\Pr\{\alpha_{\{K,j_K\}} \text{ is not chosen at time } t\} \leq (1 - (p(0) - t\Delta)^K).$$

Consider the first t iterations. Using these upper bounds, the probability that $\alpha_{\{K,j_K\}}$ is chosen at most M times among the t iterations has the following upper bound:

$$\Pr\{Y^t_{\{K,j\}} \leq M\} = \sum_{l=1}^{M} \Pr\{Y^t_{\{K,j\}} = l\} \leq \sum_{l=1}^{M} \binom{t}{l} (1)^l \Psi^{t-l}$$

where $\Psi = 1 - (p(0) - t\Delta)^K$. To show that a sum of M terms is less than δ, it is sufficient to make each element of the sum is less than (δ/M). Let us consider the case of $l = m$, where the mth term times M should be less than δ. Consequently, we need to show that $M\binom{t}{l}(1)^m(1 - (p(0) - t\Delta)^K)^{t-m}$ is bounded by δ. We see that $\binom{t}{l} \leq t^m$, and we need to show that $Mt^m\Psi^{t-m} \leq \delta$. In order to achieve this, $(1 - (p(0) - t\Delta)^K)$ must be strictly less than unity. In order for $(1-(p(0)-t\Delta)^K)$ to be less than unity, we must have $p(0) - t\Delta > 0$, which leads to the bound of Δ, i.e., $\Delta < (p(0)/t) = (1/2t)$. Hence, $(1 - (p(0) - t\Delta)^K) < 1$. By definition, $0 < \Delta$ and thus $0 < \Delta < (1/2t)$. With this value of Δ, we can simplify the analysis, and we now have $\Pr\{Y^t_{\{K,j\}} \leq M\} < Mt^m\Psi^{t-m}$, where $0 < \Psi < 1$. We now evaluate the case when $t \to \infty$ as

$$\lim_{t\to\infty} Mt^m\Psi^{t-m} = M \lim_{t\to\infty} \frac{t^m}{(1/\Psi)^{t-m}}.$$

By using L'Hospital's Rule, it follows that:

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

Therefore, for every leaf action $\alpha_{\{K,j\}}$, there exists $t = t(j)$ such that $\Pr(Y^t_{\{K,j\}} \leq M) \leq \delta$. Since $t > t(j)$ then $Y^{t(j)}_{\{K,j\}} \geq M$ gives $Y^t_{\{K,j\}} \geq M$, and therefore $\Pr\{Y^t_{\{K,j\}} \geq M\} \geq \Pr\{Y^{t(j)}_{\{K,j\}} \geq M\}$. Consequently, $\Pr\{Y^t_{\{K,j\}} \leq M\} \leq \delta$ for all $t > t(j)$. This implies that the probability of an action not being chosen as $t \to \infty$, given the restriction of Δ, is zero.

To complete the proof, let $t_0 = \max_{1\leq j\leq 2^K}\{t(j)\}$. Then for all $t > t_0$ and for all j such that $1 \leq j \leq 2^K$, we have $\Pr\{Y^t_{\{K,j\}} \leq M\} \leq \delta$. Theorem 1 is thus proven. ■

B. Marginality Property

For the second part of the proof, we need to show that, given that each action $\alpha_{\{K,j\}}$ is selected a sufficiently large number of times, the reward estimate of the optimal action will remain the largest with sufficiently large probability along the optimal path throughout the hierarchical tree structure. We first establish a baseline for Theorem 2. Let $q_{\{k,j^*_k\}}$ be the probability that the reward estimate, $\hat{d}_{\{K,j^*_K\}}$, of the optimal action, is the largest among all actions of the tree at $\mathcal{A}_{\{k,j^*_k\}}$, where $*$ is used to indicate the action that corresponds to the path of the optimal action. This relates to the various depths of the HDPA, when $K > 3$, as follows.

1) *The First Level (Root):* At this depth, we have a single LA $\mathcal{A}_{\{0,1\}}$, and $q_{\{0,1\}}$ is the probability that $\hat{d}_{\{K,j^*_K\}}$ is the maximum among all the 2^K actions.
2) *The Second Level:* $q_{\{1,j^*_1\}}$ is the probability that $\hat{d}_{\{K,j^*_K\}}$ is the maximum among all actions of the tree rooted from $A_{\{1,j^*_1\}}$. There are $2^{(K-1)}$ actions that compete for having the best reward estimate at this depth.
3) *The Interior Level(s):* For the interior depths of the hierarchy, this phenomenon holds as we follow the path down the tree at every level, having fewer actions competing for the best reward estimate as we go further down the tree.
4) *The Last/Leaf Level:* $q_{\{K-1,j^*_{K-1}\}}$ is the probability that $\hat{d}_{\{K,j^*_K\}}$ is maximum among the two actions of LA $\mathcal{A}_{\{K-1,j^*_{K-1}\}}$ at this level. There are exactly two actions that compete for having the best reward estimate here.

Theorem 2: Given a value of $\Delta \in (0,1)$, there exists a time instant denoted by $t_0 < \infty$, such that

$$q_{\{k,j_k^*\}} > 1-\delta$$

which is true $\forall t > t_0$ and $\forall k \in \{0,1,\ldots,K-1\}$.

Proof: To prove Theorem 2, we use the aspect that $q_{\{0,0\}} < q_{\{1,j_1^*\}} < \cdots < q_{\{K-1,j_{K-1}^*\}}$, since the probability of being the best from a set of actions is less than the probability of being the best from a subset of actions. Consequently, proving Theorem 2 can be achieved by proving that $q_{\{0,0\}} > 1-\delta$. Given that Theorem 1 is proven, the proof of Theorem 2 becomes identical to the corresponding proof for the DPA given in [4], [29] and [34], respectively. The additional details of the proof are thus omitted here to avoid repetition. ■

C. Submartingale Property

In order to conclude the proof in Section IV-C, we first need to show that the HDPA has the submartingale property. To do this, we now show that after a time instant t_0, the probability of choosing the optimal action is increasing, *in the expected sense*. This feature is different from the probability of being monotonically increasing, which is a very strong condition.

Theorem 3: Under the HDPA, the quantity

$$\left\{p_{\{k,j_k^*\}}(t)\right\}$$

where $k \in \{1,\ldots,K\}$ and $t > t_0$ is a submartingale.

Proof: We first formalize the submartingale property by denoting a sequence of random variables as $X_1, X_2, \ldots, X_t$. The sequence is a submartingale if for any time instant t

$$E[X_t] < \infty \text{ and } E[X_{t+1}|X_1,X_2,\ldots,X_t] \geq X_t.$$

To prove Theorem 3, we first observe that $p_{\{k,j_k^*\}}$ is a probability, which means $p_{\{k,j_k^*\}} \leq 1 < \infty$. Second, we explicitly calculate $E[p_{\{k,j_k^*\}}(t)]$, for all $k \in \{1,\ldots,K\}$. The proof of this theorem is achieved by an inductive argument. At every level, we consider the nodes that are one and two levels below it, respectively. Indeed, this is true because that is the structure that effectively remains at the specific node, and is essentially depicted in Fig. 1, where the root node of the subtree is determined by the decision of two children and four grandchildren. Once the proof has been proven for such a scenario, the overall proof follows trivially, because the decision of every node is based on the decision of its immediate *two* children and *four* grandchildren. A straightforward inductive argument formalizes this to be true for the overall tree, since it is true at every level for the subtree of depth two from the root of the corresponding subtree. Thus, in what follows, we merely use the tree structure of Fig. 1 to formalize the proof.

We look at the first three levels of the LA hierarchy and calculate $E[p_{\{k,j_k^*\}}(t)]$, when $k = 2$. Without loss of generality, we simplify the notations to what are shown in Fig. 1, and let $\mathcal{A}_{\{1,1\}}$ ($\alpha_{\{1,1\}}$) and $\mathcal{A}_{\{2,1\}}$ ($\alpha_{\{2,1\}}$) be the LA (actions) on the optimal path. One can see that this implies that the optimal path is the one consisting of all the leftmost nodes in both levels. We denote the set of all the action probabilities in time instant t in the tree as $\mathcal{P}(t)$. Then, by going through all the four paths that lead to four individual actions at level 2, and by following the action probability updating rules of the HDPA, we can calculate the expected probability of choosing the optimal action at level 2 as

$$\begin{aligned}
&E[p_{21}(t+1)|\mathcal{P}(t)]\\
&\quad= p_{11}p_{21}(d_{21}(q_{11}(p_{21}+\Delta)+(1-q_{11})(p_{21}-\Delta))\\
&\qquad+(1-d_{21})p_{21})\\
&\qquad+p_{11}p_{22}(d_{22}(q_{11}(p_{21}+\Delta)+(1-q_{11})(p_{21}-\Delta))\\
&\qquad+(1-d_{22})p_{21})\\
&\qquad+p_{12}p_{23}p_{21}+p_{12}p_{24}p_{21}\\
&\quad= p_{11}p_{21}(d_{21}q_{11}p_{21}-d_{21}q_{11}\Delta+d_{21}p_{21}-d_{21}\Delta\\
&\qquad-d_{21}q_{11}p_{21}+d_{21}q_{11}\Delta+p_{21}-d_{21}p_{21})\\
&\qquad+p_{11}p_{22}(d_{22}q_{11}p_{21}-d_{22}q_{11}\Delta+d_{22}p_{21}-d_{22}\Delta\\
&\qquad-d_{22}q_{11}p_{21}+d_{22}q_{11}\Delta+p_{21}-d_{22}p_{21})\\
&\qquad+p_{12}p_{21}\\
&\quad= p_{11}p_{21}d_{21}\Delta(2q_{11}-1)+p_{11}p_{21}p_{21}\\
&\qquad+p_{11}p_{22}d_{22}\Delta(2q_{11}-1)+p_{11}p_{22}p_{21}\\
&\qquad+p_{12}p_{21}\\
&\quad= p_{11}\Delta(p_{21}d_{21}+p_{22}d_{22})(2q_{11}-1)+p_{21}.
\end{aligned}$$

Note that in the above expression, we have omitted all the time instant information to keep the expression simpler and neat. The complete version should be

$$\begin{aligned}
&E[p_{21}(t+1)|\mathcal{P}(t)]\\
&\quad= p_{11}(t)\Delta\Big(p_{21}(t)d_{21}+p_{22}(t)d_{22}\Big)\Big(2q_{11}(t)-1\Big)+p_{21}(t).
\end{aligned}$$

Following the same manner in which we calculated the above $E[p_{21}(t+1)|\mathcal{P}(t)]$, we are able to get the generalized formula for the expected probability of choosing the optimal action at level k

$$\begin{aligned}
&E\big[p_{\{k,j_k^*\}}(t+1)|\mathcal{P}(t)\big]\\
&\quad= p_{\{k,j_k^*\}}(t)+\left(\prod_{l=1,\ldots,k-1}p_{\{l,j_l^*\}}(t)\right)\\
&\qquad\times\left(\sum_{j=1,2}p_{\{k,j\}}(t)d_{\{k,j\}}\Delta(2q_{\{k-1,j_{k-1}^*\}}(t)-1)\right)
\end{aligned}$$

which can be proven by an inductive argument. The difference between $E[p_{\{k,j_k^*\}}(t+1)]$ and $p_{\{k,j_k^*\}}(t)$ is thus

$$\begin{aligned}
\mathrm{Diff}_{p_{\{k,j_k^*\}}(t)} &= E[p_{\{k,j_k^*\}}(t+1)|P(t)]-p_{\{k,j_k^*\}}(t)\\
&= \left(\prod_{l=1,\ldots,k-1}p_{\{l,j_l^*\}}(t)\right)\\
&\quad\times\left(\sum_{j=1,2}p_{\{k,j\}}(t)d_{\{k,j\}}\Delta(2q_{\{k-1,j_{k-1}^*\}}(t)-1)\right).
\end{aligned}$$

As $p_{\{k,j\}}(t) > 0$ and $\hat{d}_{\{k,j\}}(t) > 0$ for all t, k, and j, we clearly see that if $\forall t > t_0$, $q_{\{k,j_k^*\}}(t) > (1/2)$, then $\mathrm{Diff}_{p_{\{k,j_k^*\}}(t)} > 0$, and the sequence $p_{\{k,j_k^*\}}$ with $t > t_0$ is a submartingale. Therefore, we only need to let $1-\delta = (1/2)$, then, by Theorem 2, there exists a time instant t_0, such that for all $t > t_0$ we have

$$q_{\{k,j_k^*\}}(t) > \frac{1}{2}.$$

Theorem 3 is thus proven. ■

D. ϵ-Optimality of the HDPA

Finally, we show the ϵ-Optimality of the HDPA.

Theorem 4: For all stationary stochastic Environments, the HDPA is ϵ-optimal, i.e., the HDPA will converge to the optimal path $\mathcal{P}^* = \{p_{\{k,j_k^*\}}\}, k = \{1,\ldots,K\}$. Formally, given any $1-\delta > (1/2)$, there exists a $\Delta \in (0,1)$ and a time instant $t_0 < \infty$, such that for all $\Delta < \Delta_0$ and for all time instants $t > t_0$, we have $q_{\{k,j_k^*\}}(t) > 1-\delta, \forall k$, and the quantity

$$\Pr\left\{p_{\{k,j_k^*\}}(\infty) = 1\right\} \to 1, \quad \text{where } k \in \{1,\ldots,K-1\}.$$

Proof: We can interpret Theorem 4 as follows. If the probability of choosing the optimal action in each level converges to unity, then the entire tree will converge to the optimal path, which consists of the optimal nodes from each level. We thus can follow the level-wise approach to prove Theorem 4.

We again refer to the simplified tree and notations in Fig. 1 and consider the first two levels in the LA hierarchy, i.e., when $k = 1$. We are to prove that $\Pr\{p_{11}(\infty) = 1\} \to 1$, and the proof is essentially the same as how we proved that a flat DPA converges to the optimal action in [36]. Consequently, the proof can be based on the submartingale convergence theory, and we can utilize a Regular function to indirectly study the convergence probability. First, as per Theorem 3, $p_{11}(\infty)$ is a submartingale. According to the submartingale convergence theory, $p_{11}(\infty) = 0$ or 1. Second, the convergence probability $\Pr\{p_{11}(\infty) = 1\}$ can be written as $\Pr\{P_{01}(\infty) = e_m\}$, where e_m is the unit vector and m is the index of element which corresponds to the optimal action. At node $\mathcal{A}_{\{0,1\}}$, $e_m = e_1 = [1,0]$, as the optimal action is the first action. Therefore, proving $\Pr\{p_{11}(\infty) = 1\} \to 1$ is equivalent to proving $\Gamma(P_{01}) = \Pr\{P_{01}(\infty) = e_1\} \to 1$.

To prove this, we shall use the theory of Regular functions, and follow an argument similar to the one given in [36]. Let $\Phi(P)$ be a function of P, and we define an operator U as

$$U\Phi(P) = E[\Phi(P(t+1))|P(t) = P].$$

Then, we repeatedly apply U, resulting in the expression

$$U^t\Phi(P) = E[\Phi(P(t))|P(0) = P].$$

If $\Phi(P) = U\Phi(P) = U^2\Phi(P) = \cdots = U^\infty\Phi(P)$, we call $\Phi(P)$ a regular function of P. If $\Phi(P) \geq U\Phi(P) \geq U^2\Phi(P) \geq \cdots \geq U^\infty\Phi(P)$, $\Phi(P)$ is a super-regular function of P, and when $\Phi(P) \leq U\Phi(P) \leq U^2\Phi(P) \leq \cdots \leq U^\infty\Phi(P)$, $\Phi(P)$ is a sub-regular function of P. Furthermore, if $\Phi(P)$ satisfies the boundary conditions

$$\Phi(e_m) = 1 \text{ and } \Phi(e_j) = 0, (\text{for } j \neq m) \tag{1}$$

then, as per the definition of Regular functions and the submartingale convergence theory, we have

$$\begin{aligned} U^\infty\Phi(P) &= E[\Phi(P(\infty))|P(0) = P] \\ &= \sum_{j=1,2}\Phi(e_m)\Pr\{P(\infty) = e_j|P(0) = P\} \\ &= \Pr\{P(\infty) = e_m|P(0) = P\} \\ &= \Gamma(P). \end{aligned} \tag{2}$$

Equation (2) shows that $\Gamma(P)$ is exactly the function $\Phi(P)$ upon which U is applied an infinite number of times, and this function can be lower/upper bounded by $\Phi(P)$ if $\Phi(P)$ is a sub-regular/super-regular.

Our goal is to find a proper sub-regular function to serve as the lower bound $\Gamma(P_{01})$. We solve this by first finding a corresponding super-regular function of P_{01}. Let us consider a specific instantiation of Φ to be the function Φ_1 and

$$\Phi_1(P_{01}) = e^{-x_1 p_{11}}$$

where x_1 is a positive constant. It follows that, under the HDPA, we have

$$\begin{aligned} &U(\Phi_1(P_{01})) - \Phi_1(P_{01}) \\ &\quad= E[\Phi_1(P_{01}(t+1))|\mathcal{P}(t)] - \Phi_1(P_{01}(t)) \\ &\quad= E\left[e^{-x_1 p_{11}(t+1)}|P_{01}(t)\right] - e^{-x_1 p_{11}(t)} \\ &\quad= \sum_{j=1,2} e^{-x_1(p_{11}+\Delta)}p_{1j}d_{1j}q_{01} \\ &\qquad+ \sum_{j=1,2} e^{-x_1(p_{11}-\Delta)}p_{1j}d_{1j}(1-q_{01}) \\ &\qquad+ \sum_{j=1,2} e^{-x_1 p_{11}}p_{1j}(1-d_{1j}) - e^{-x_1 p_{11}} \\ &\quad= \sum_{j=1,2} p_{1j}d_{1j}e^{-x_1 p_{11}}\bigg(q_{01}\Big(\left(e^{-x_1\Delta} - e^{x_1\Delta}\right) \\ &\qquad\qquad + (e^{x_1\Delta} - 1)\Big)\bigg). \end{aligned} \tag{3}$$

In the above equation, time instant (t) has been omitted from $p_{1j}(t)$, $p_{11}(t)$, and $q_{01}(t)$. We now need to determine a proper value for x_1 such that $\Phi_1(P)$ is super-regular, i.e.,

$$U(\Phi_1(P_{01})) - \Phi_1(P_{01}) \leq 0$$

which is equivalent to solving the following inequality:

$$q_{01}\left(e^{-x_1\Delta} - e^{x_1\Delta}\right) + \left(e^{x_1\Delta} - 1\right) \leq 0.$$

By Taylor expansion, the above equation can be approximated

$$\begin{aligned} &q_{01}\left(1 - x_1\Delta + \frac{x_1^2\Delta^2}{2} - 1 - x_1\Delta - \frac{x_1^2\Delta^2}{2}\right) \\ &\qquad + 1 + x_1\Delta + \frac{x_1^2\Delta^2}{2} - 1 \leq 0 \\ &\quad\Rightarrow x_1\left(\frac{x_1\Delta}{2} + 1 - 2q_{01}\right) \leq 0 \\ &\quad\Rightarrow 0 \leq x_1 \leq \frac{2(2q_{01}-1)}{\Delta}. \end{aligned} \tag{4}$$

Let us now introduce a new function, $\phi_1(P_{01})$, which satisfies the boundary conditions as follows:

$$\phi_1(P_{01}) = \frac{1 - e^{-x_1 p_{11}}}{1 - e^{-x_1}} = \begin{cases} 1, & \text{when } P_{01} = e_1 \\ 0, & \text{when } P_{01} = e_2 \end{cases}$$

where x_1 is similar to how it is defined in $\Phi_1(P_{01})$. It follows that, if $\Phi_1(P_{01}) = e^{-x_1 p_{11}}$ is super-regular (sub-regular), then $\phi_1(P_{01}) = (1 - e^{-x_1 p_{11}}/1 - e^{-x_1})$ is a sub-regular (super-regular) [2]. The definition of x_1 that renders $\Phi_1(P_{01})$ to be super-regular makes $\phi_1(P_{01})$ to be sub-regular. Following the property of Regular functions, it follows that:

$$\Gamma(P_{01}) \geq \phi_1(P_{01}) = \frac{1 - e^{-x_1 p_{11}}}{1 - e^{-x_1}}. \tag{5}$$

As (5) holds for every x_1 bounded by (4), we can choose the largest value $x_{1\max} = (2(2q_{01}-1)/\Delta)$. When $\Delta \to 0$, we have $x_{1\max} \to \infty$, which renders $\phi_1(P_{01}) \to 1$, hence $\Gamma(P_{01}) \to 1$. Thus, $\Pr\{p_{11}(\infty) = 1\} \to 1$ under the HDPA.

The above proof methodology can be applied to higher levels of the HDPA hierarchy. Take the simplest hierarchical DPA with $K = 2$ for an example, again, we refer to Fig. 1 for the simplified notations, and we are to prove that $\Gamma(P_{11}) = \Pr\{P_{11}(\infty) = e_1\} = \Pr\{p_{21}(\infty) = 1\} \to 1$.

Let us consider another specific instantiation of Φ to be

$$\Phi_2(P_{11}) = e^{-x_2 p_{21}}$$

where x_2 is a positive constant. Under the HDPA, we have

$$\begin{aligned}
&U(\Phi_2(P_{11})) - \Phi_2(P_{11}) \\
&= E[\Phi_2(P_{11}(t+1))|\mathcal{P}(t)] - \Phi_2(P_{11}(t)) \\
&= E\big[e^{-x_2 p_{21}(t+1)}|\mathcal{P}(t)\big] - e^{-x_2 p_{21}(t)} \\
&= e^{-x_2(p_{21}+\Delta)}(p_{11}p_{21}d_{21}q_{11}) \\
&\quad + e^{-x_2(p_{21}-\Delta)}(p_{11}p_{21}d_{21}(1-q_{11})) \\
&\quad + e^{-x_2 p_{21}}(p_{11}p_{21}(1-d_{21})) + e^{-x_2(p_{21}+\Delta)}(p_{11}p_{22}d_{22}q_{11}) \\
&\quad + e^{-x_2(p_{21}-\Delta)}(p_{11}p_{22}d_{22}(1-q_{11})) \\
&\quad + e^{-x_2 p_{21}}(p_{11}p_{22}(1-d_{22})) + e^{-x_2 p_{21}}(p_{12}p_{23}) \\
&\quad + e^{-x_2 p_{21}}(p_{12}p_{24}) - e^{-x_2 p_{21}} \\
&= e^{-x_2 p_{21}}\Big[p_{11}p_{21}\big(e^{-x_2\Delta}d_{21}q_{11} + e^{x_2\Delta}d_{21}(1-q_{11})\big) + (1-d_{21}) \\
&\qquad + p_{11}p_{22}\big(e^{-x_2\Delta}d_{22}q_{11} + e^{x_2\Delta}d_{22}(1-q_{11}) \\
&\qquad + (1-d_{22})) + p_{12}\Big] - e^{-x_2 p_{21}} \\
&= e^{-x_2 p_{21}}\Bigg[p_{11}\Bigg(\sum_{j=1,2} p_{2j}\big(e^{-x_2\Delta}d_{2j}q_{11} + e^{x_2\Delta}d_{2j}(1-q_{11}) \\
&\qquad + (1-d_{2j})\big)\Bigg) + p_{12}\Bigg] - e^{-x_2 p_{21}} \\
&= e^{-x_2 p_{21}}\Bigg[p_{11}\Bigg(\sum_{j=1,2} p_{2j}\big(e^{-x_2\Delta}d_{2j}q_{11} + e^{x_2\Delta}d_{2j}(1-q_{11}) \\
&\qquad - d_{2j})\big)\Bigg) + p_{11}\sum_{j=1,2} p_{2j} + p_{12}\Bigg] \\
&\quad - e^{-x_2 p_{21}} \\
&= e^{-x_2 p_{21}}\Bigg[p_{11}\Bigg(\sum_{j=1,2} p_{2j}\big(e^{-x_2\Delta}d_{2j}q_{11} + e^{x_2\Delta}d_{2j}(1-q_{11}) \\
&\qquad - d_{2j})\big)\Bigg) + 1\Bigg] - e^{-x_2 p_{21}} \\
&= e^{-x_2 p_{21}}\Bigg[p_{11}\Bigg(\sum_{j=1,2} p_{2j}d_{2j}\big(e^{-x_2\Delta}q_{11} + e^{x_2\Delta}(1-q_{11}) - 1\big)\Bigg)\Bigg] \\
&= e^{-x_2 p_{21}}\Bigg[p_{11}\Bigg(\sum_{j=1,2} p_{2j}d_{2j}\big(q_{11}\big(e^{-x_2\Delta} - e^{x_2\Delta}\big) \\
&\qquad + (e^{x_2\Delta} - 1)\big)\Bigg)\Bigg]. \qquad (6)
\end{aligned}$$

Just as in (3), in the above (6), the time instant (t) has been omitted from $p_{2j}(t)$, $p_{11}(t)$, and $q_{11}(t)$, to make the notation less cumbersome. In order for $\Phi_2(P_{11})$ to be super-regular, we need

$$\begin{aligned}
&U(\Phi_2(P_{11})) - \Phi_2(P_{11}) \le 0 \\
&\quad \Rightarrow q_{11}\big(e^{-x_2\Delta} - e^{x_2\Delta}\big) + \big(e^{x_2\Delta} - 1\big) \le 0 \\
&\quad \Rightarrow 0 \le x_2 \le \frac{2(2q_{11}-1)}{\Delta}. \qquad (7)
\end{aligned}$$

The same x_2 will render $\phi_2(P_{11}) = (1 - e^{-x_2 p_{21}}/1 - e^{-x_2})$ to be sub-regular. Clearly, $\phi_2(P_{11})$ meets the boundary condition, thus

$$\Gamma(P_{11}) \ge \phi_2(P_{11}) = \frac{1 - e^{-x_2 p_{21}}}{1 - e^{-x_2}}. \qquad (8)$$

Similarly, when $\Delta \to 0$, we have $x_{2\max} = (2(q_{11}-1)/\Delta) \to \infty$, which renders $\phi_2(P_{11}) \to 1$, whence $\Gamma(P_{11}) = \Pr\{p_{21}(\infty) = 1\} \to 1$ under the HDPA.

As explained above, the overall proof of the entire tree follows by a simple inductive argument. Defining x_k as a positive constant, we can generalize (6) to any level k

$$\begin{aligned}
&U(\Phi_k(P_{\{k-1,j^*_{k-1}\}})) - \Phi_k(P_{\{k-1,j^*_{k-1}\}}) \\
&\quad = E[\Phi_k(P_{\{k-1,j^*_{k-1}\}}(t+1))|\mathcal{P}(t)] - \Phi_k(P_{\{k-1,j^*_{k-1}\}}(t)) \\
&\quad = E\Big[e^{-x_k p_{\{k,j^*_k\}}(t+1)}|\mathcal{P}(t)\Big] - e^{-x_k p_{\{k,j^*_k\}}(t)} \\
&\quad = e^{-x_k p_{\{k,j^*_k\}}}\Bigg(\prod_{l=0,\ldots,k-1} p_{\{l,j^*_l\}}(t)\Bigg) \\
&\qquad \times \Bigg(\sum_{j=1,2} p_{\{k,j_k\}}d_{\{k,j\}}\big(q_{\{k-1,j^*_{k-1}\}}\big(e^{-x_k\Delta} - e^{x_k\Delta}\big) \\
&\qquad + (e^{x_k\Delta} - 1)\big)\Bigg) \qquad (9)
\end{aligned}$$

and the conclusion is the same: when $\Delta \to 0$, we have $x_{\mathrm{kmax}} = (2(2q_{\{k-1,j^*_{k-1}\}} - 1)/\Delta) \to \infty$, which renders $\phi_k(P_{\{k-1,j^*_{k-1}\}}) \to 1$. Hence $\Gamma(P_{\{k-1,j^*_{k-1}\}}) = \Pr\{p_{\{k,j^*_k\}}(\infty) = 1\} \to 1$ under the HDPA.

As the LA converges to the optimal action at each level, the HDPA converges the optimal path, proving Theorem 4. ■

V. Numerical Results

To demonstrate the performance of the proposed HDPA scheme, we carried out extensive simulations for different Environments with “many” actions. To ensure the credibility of our simulations, we increased the number of experiments and the convergence criteria compared to the experiments in [31]. As highlighted earlier, one of the drawbacks of the HCPA is that it has a sluggish increase in updating the action probabilities when the probability to be increased approaches unity. As opposed to this, we expected that the HDPA, which has a constant increment in the action probability, would have significantly faster convergence than that of the HCPA when the convergence criterion is close to unity. Our results presented below, demonstrated that the HDPA, indeed, required significantly fewer iterations for cases requiring high accuracy.

TABLE I
BENCHMARK ACTION REWARD PROBABILITIES ESTABLISHED IN [31]

j_K	Ω	j_K	Ω	j_K	Ω	j_K	Ω	j_K	Ω	j_K	Ω	j_K	Ω	j_K	Ω
1	0.3934	9	0.0333	17	0.7362	25	0.02152	33	0.6214	41	0.0970	49	0.2413	57	0.4763
2	0.9902	10	0.4323	18	0.7603	26	0.2399	34	0.9777	42	0.1319	50	0.1714	58	0.4446
3	0.4883	11	0.6926	19	0.5142	27	0.7509	35	0.4232	43	0.1738	51	0.8512	59	0.9617
4	0.5768	12	0.3474	20	0.2273	28	0.8773	36	0.02773	44	0.8901	52	0.9791	60	0.0329
5	0.2023	13	0.6152	21	0.6080	29	0.4962	37	0.1255	45	0.3511	53	0.7443	61	0.5004
6	0.2390	14	0.0900	22	0.4791	30	0.5649	38	0.5650	46	0.8945	54	0.3469	62	0.3784
7	0.5887	15	0.0850	23	0.9339	31	0.9202	39	0.1660	47	0.6133	55	0.8707	63	0.6553
8	0.8894	16	0.5652	24	0.3808	32	0.1335	40	0.0148	48	0.4813	56	0.3863	64	0.9737

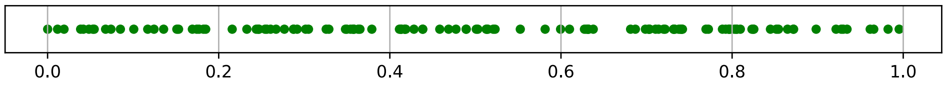


Fig. 2. Action reward probabilities for the 128 actions environment.

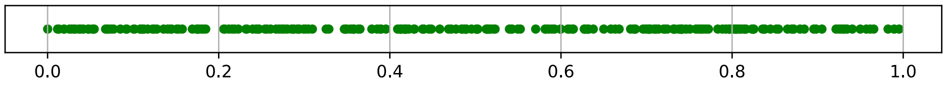


Fig. 3. Action reward probabilities for the 256 actions environment.

A. Simulation Environments

We conducted simulations for Environments with 16, 32, 64, 128, 256, and 512 actions. We configured the simulation Environments for the 16, 32, and 64 actions on the benchmark action reward probabilities established in [31], which are listed in Table I. The table lists 64 reward probabilities. The parameter j_K in Table I indicates the index of the action at depth K in the HDPA structure, and $\Omega = \Pr\{\beta = 0\}$ shows the probability of that action obtaining a Reward from the Environment. We utilized the first 16 probabilities as the 16-action Environment, i.e., $j_K \in \{1, \ldots, 16\}$. Likewise, for the 32-action Environment, we used $j_K \in \{1, \ldots, 32\}$, and so on. For the 128 and 256 actions Environments, we uniformly generated 128 and 256 reward probabilities between zero and unity, visualized in Figs. 2 and 3, respectively. Note that for 128, 256, and 512 actions, the reward probabilities utilized were distinct from those used for the Environments in [31].

B. Algorithms' Learning Parameters

From the mathematical proof above and the established theory of VSSA, we know that when the learning parameter, Δ, is sufficiently small, the HDPA will converge to the action that has the maximum reward probability with a probability close to unity. The same is true when it concerns the value of λ for the HCPA. The respective values for λ and Δ determine how quickly the LA achieves convergence. The higher the learning parameters are configured, the faster the algorithms converge. However, if the learning parameters are configured too high, the algorithm might not converge to the optimal action with high probability. Therefore, the tuning of these parameters is a trade-off between the accuracy of finding the optimal action and the speed of convergence.

To find the best values for λ and Δ, we utilized a top-down approach.[9] More specifically, we decreased the value of the learning parameters in a step-wise manner with two decimals precision until their configured values made the LA achieve 100% accuracy in converging to the optimal action for all the number of experiments arranged. Consequently, the values of the learning parameters that fulfilled these criteria represented the assumed "best" values for λ and Δ given the distinct Environments used in the simulations. We emphasize that tuning the values of λ and Δ is challenging because the Environments's stochastic nature can cause uncertainty in the values of λ and Δ. Moreover, although we obtained the values for λ and Δ through extensive testing, it is not true that the HDPA will "always" select the optimal action because the convergence is, indeed, in probability due to the ϵ-optimal property. The reader should also note that the λ and Δ values are dependent on the Environment's reward probabilities and that the best values of λ and Δ vary from Environment to Environment. Therefore, we refer to the values determined for λ and Δ in this article as the "best" learning parameters for the given Environments, and not the "optimal" ones.

[9]This issue is application-dependent, and there is no hard-and-fast rule to determine this. We thank the anonymous Referee who pointed this out to us.

C. Average Number of Iterations

The average number of iterations required before *convergence* (to a convergence threshold, T) is an established parameter for evaluating a learning scheme's. efficiency.

We first address the simulation results presented in Table II. For these simulations, we conducted 600 experiments and configured the convergence criterion to be 0.992. Thus, we affirmed that the LA had converged when it achieved a 0.992 probability of choosing one of the actions in its action probability vector. All the results presented in Table II were based on *all 600* experiments converging to the optimal action.

For the Environment with 16 actions, the "best" learning parameters were $\lambda = 0.0043$ and $\Delta = 0.0011$. For the Environment with 32 and 64 actions, the best learning parameters were $\lambda = 0.00057$ $\lambda = 3.6\mathrm{e}^{-5}$, $\Delta = 0.00015$ and $\Delta = 9.9\mathrm{e}^{-6}$, respectively. The "best" obtained values for the Environment with 128 actions were $\lambda = 3.9\mathrm{e}^{-5}$ and $\Delta = 9.7\mathrm{e}^{-6}$. For 256 actions, we obtained $\lambda = 5.9\mathrm{e}^{-6}$ and $\Delta = 1.5\mathrm{e}^{-6}$ as the "best" learning parameters. For the 512-action case, visualized in Fig. 8, we used $\lambda = 7.9\mathrm{e}^{-5}$ and $\Delta = 1.7\mathrm{e}^{-5}$. From these, we can observe that the optimal action is further away from the

TABLE II
SIMULATION RESULTS OBTAINED FOR THE VARIOUS ENVIRONMENTS

Nr. of Actions	HCPA		HDPA	
16	Mean:	1,366.61	Mean:	868.25
	Std:	121.14	Std:	135.50
32	Mean:	10,281.84	Mean:	6,172.38
	Std	681.82	Std:	744.84
64	Mean:	169,839.67	Mean:	100,638.41
	Std:	13,687.48	Std:	17,653.41
128	Mean:	155,088.62	Mean:	97,795.59
	Std:	10,613.21	Std:	13,266.12
256	Mean:	1,039,215.58	Mean:	632,985.27
	Std:	88,744.96	Std:	70,978.51
512	Mean:	95,354.61	Mean:	78,779.84
	Std:	13,099.73	Std:	15,418.82

suboptimal one. Therefore, the Environment is "simpler" than, e.g., the 64, 128, and 256-action cases, because the algorithms required less number of iterations for this Environment. Thus, the hardness of the Environment can impact the number of iterations more than the number of actions.

Table II includes both algorithms' results, and lists both the Mean and the Standard Deviation (Std) of the number of iterations required before convergence. Let us first consider the cases with 16, 32, and 64 actions based on the benchmark probabilities in Table I. The HDPA required just 64%, 60%, and 59% of the total number of iterations that the HCPA required for the 16, 32, and 64 actions, respectively. 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 three environments.

Considering the results for 128 actions (with action reward probabilities depicted in Fig. 2), the HDPA converged within approximately 97 800 iterations on average, while the HCPA required 155 000 iterations. Comparing the algorithms' Std, we again observe that the HDPA had more variation compared with that of the HCPA. This indicates that HDPA is slightly more unpredictable in its convergence iterations. As we can observe for the 256 case, the HDPA required significantly fewer iterations than the HCPA. More specifically, the HDPA achieved convergence with just 60% of the iterations that the HCPA required, which clearly demonstrated the advantage of the HDPA over HCPA in the above-studied configurations.

D. Illustrative Details of Convergence Analysis

In this section, we present more convergence details for the Environments with 64 actions. For these simulations, we conducted 1000 experiments with 0.999 as the convergence criterion. The "best" learning parameters for these simulations, found with the top-down approach, were $\lambda = 3.8e^{-5}$ and $\Delta = 9.4e^{-6}$, respectively. In Fig. 4, a cumulative representation of the number of iterations required with the different schemes is presented. As we can observe, the HDPA required significantly fewer iterations, and we see that approximately 90% of the experiments converged within approximately 140 000 iterations. In comparison, none of the HCPA experiments required less than 200 000 iterations.

In Fig. 5, we present the number of iterations required for the HCPA and the HDPA through a scatter representation. We can observe that some of the HDPA experiments required more iterations than the HCPA, and that some of them coincide

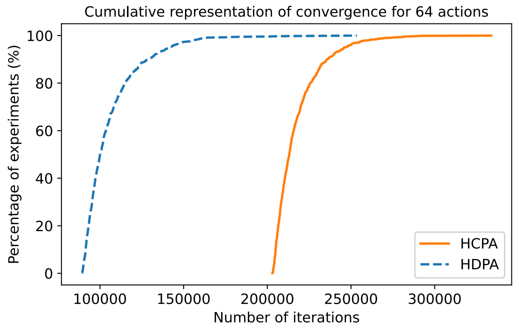


Fig. 4. Cumulative representation of the percentage of experiments converging within a certain number of iterations. The figure is based on 1000 experiments, with 0.999 as the convergence criterion for 64 actions.

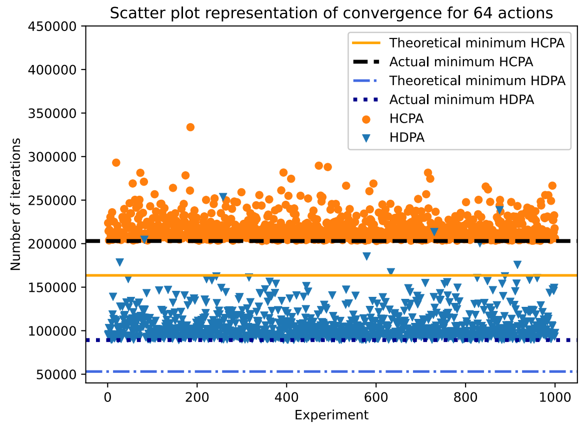


Fig. 5. Scatter plot of the experiment number and the number of iterations it took before the LA converged for that experiment. The figure is based on 1000 experiments, with 0.999 as the convergence criterion for 64 actions.

with the number of iterations as the HCPA. However, these are outliers, and most of the experiments are concentrated at around 100 000 for the HDPA. For the HCPA we see that the number of iterations is located at around 200 000. The reason that the iteration numbers in most of the experiment instances is close to the actual minimum number is that the numbers of iterations become close to each other after the action selection probabilities reach a certain level. In these cases, it is very likely that the LA choose the optimal actions, and that they also obtain a reward for choosing these actions. Due to the pursuit concept, the system rewarded the current best actions even if a suboptimal action was chosen and rewarded.

E. Performance for Different Convergence Criteria

In Figs. 6 and 7, we present the performance of the HCPA and the HDPA for different convergence criteria for 64 actions. In these simulations, we conducted 100 experiments, where we used the top-down approach for finding the "best" λ and Δ for each individual convergence criterion. The "best" λ and Δ for the convergence criteria between 0.90 and 0.99 can be made available to the interested readers. For the convergence criteria between 0.990 and 0.999, the "best" learning parameters were found to be $\lambda = 5.3e^{-5}$ and $\Delta = 1.0e^{-5}$ for all convergence criteria.[10] The reason why these learning parameters remained

[10]The reader should note that the "best" values for λ and Δ found here differ from the "best" ones reported in Sections V-C and V-D because the numbers of experiments are different. Indeed, the 1 000 experiments stipulated in Section V-D set a higher requirement to the learning parameters than the 100 experiments in Section V-E, and thus required more conservative learning.

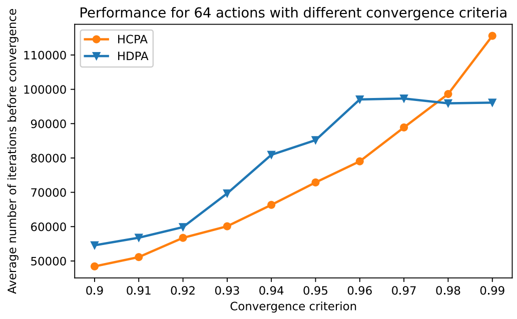


Fig. 6. Average number of iterations required before convergence for 100 experiments with different convergence criteria from 0.90 to 0.99 for a 64-action environment.

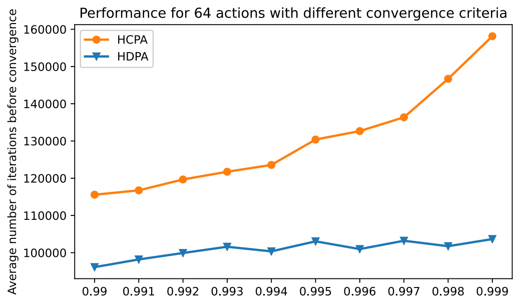


Fig. 7. Average number of iterations required before convergence for 100 experiments with different convergence criteria from 0.990 to 0.999 for a 64-action environment.

constant is that when the LA reached such a high probability level as 0.990, it rarely changed to another action than the one it was currently pursuing. Consequently, the values for λ and Δ that we found represented the highest (fastest) learning parameters for all convergence criteria above 0.990.

In Fig. 6, we can observe that the HCPA required less iterations on average for the convergence criteria between 0.9 and 0.97. However, for the convergence criteria of 0.98 and 0.99, the HDPA had fewer required iterations on average. Consequently, we observed that the HDPA was more efficient as the convergence requirement was configured closer to unity. The effectiveness of the HDPA for higher convergence criteria was further verified in Fig. 7, where the HDPA had fewer required iterations on average for all convergence requirements. Additionally, it is clear from Fig. 7 that the difference between the two schemes increased as the criterion increased.

F. Comparison With Other Competing LA

Before we conclude this article, it is pertinent that we compare our results with another set of algorithms.[11] These papers [37], [38], [39] (referred to as AlgJu1, AlgJu2, and AlgJu3) have all been written by the same team of researchers, and it is prudent to compare them quantitatively and qualitatively.

Our first remark is that these papers are brilliant within the field of LA. We have implemented all of them and found that they are very competitive even against the best-reported

[11]We were unaware of these papers, when we submitted the initial version of the paper. We are extremely grateful to the anonymous Referee who directed us to them. This section is dedicated to such a mutual comparison.

TABLE III
COMPARISON BETWEEN THE HDPA AND ALGJU3 BASED ON 1000 EXPERIMENTS

Nr. of Actions	HDPA		AlgJu3	
10 (E3 in [39])	Mean:	6,813.089	Mean:	3,721.99 (6,134.39)
	Std:	1274.38	Std:	235.14 (159.38)
	Acc:	100%	Acc:	99.2% (100%)
	Δ:	0.00026	n:	59 (100)
64 (Env. in [31])	Mean:	104,884.60	Mean:	35,713.68
	Std:	18393.40	Std:	532.42
	Acc:	100%	Acc:	100%
	Δ:	$9.5e^{-}6$	n:	100
512 (in Fig. 8)	Mean:	79,623.64	Mean:	46,004.62
	Std:	16,432.28	Std:	587.53
	Acc:	100%	Acc:	100%
	Δ:	$1.7e^{-}5$	n:	10

schemes. Of particular interest, however, is the scheme AlgJu3 presented in [39], which is noticeably superior to AlgJu1 and AlgJu2 from [37] and [38], respectively. In the interest of brevity and space, we merely report below the results comparing our work with the paper of [39], while the comparisons with [37] and [38] are included in the Ph.D. thesis [40].

The methodology that is used in AlgJu3 has the potential of making it the most superior algorithm. However, the underlying principle is a two-edged sword as explained below.

1) The principle which renders this article competitive is that after a sufficiently large number of iterations it is able to distinguish between "the boys and the men." In this way, the algorithm decides to discard many of the rather inferior actions, resulting in a brilliant paradigm of learning within a *smaller* action space. This is the reason why the method becomes so competitive.
2) The negative side of such a decision is that if such an action-pruning task is undertaken after a large number of iterations, it is, indeed, not profitable. But if it is undertaken before this "large" number of iterations, in many cases the scheme, unfortunately, discards some of the more superior actions. While this is mostly an infrequent occurrence, in reality, when the actions are highly competitive, it does occur to a noticeable extent by which some of the best actions are discarded.
3) The above negative phenomenon, while being less likely in stationary Environments, is an almost-predominant occurrence in "nonstationary" Environments.
4) While the above phenomenon can be mitigated by using a small enough learning parameter, the question of determining the size of the parameter is still open, other than by invoking a meta-learning scheme.

The results of comparing our algorithm with AlgJu3 [39] are given in Table III, where as mentioned above, the weaker schemes of AlgJu1 and AlgJu2 are omitted. The reader will observe that AlgJu3 is superior in most cases, but we have to emphasize that the results reported in the article are not as stringent as those reported here because we have required a categorical 100% accuracy. AlgJu3 is arguably superior, but it has an impediment in that in some cases it discards the best action from the entire competition. Thus, *while it appears to be superior*, as explained in the above items, the truth of the matter is that its superiority is a result of operating within a diminished action space.

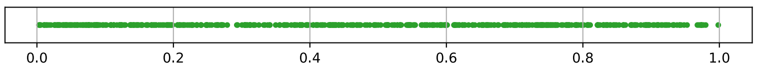


Fig. 8. Action reward probabilities for the 512-action environment.

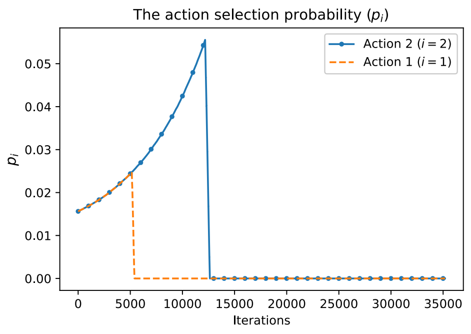


Fig. 9. Performance for a 64-action environment where the optimal action changes over time. After about 13 000 iterations both α_1 and α_2 are discarded from the competition because of the environment's nonstationarity.

This comment has a major significance as mentioned in the third item above. The graph in Fig. 9 displays the action selection probability when α_1 is the best action and after about 13 000 iterations it discards both α_1 and α_2 because of the nonstationarity of the Environment.

VI. Conclusion

In this article, we have thoroughly surveyed the strategies to enhance the speed and accuracy in the field of LA over the last six decades. By incorporating the major phenomena into a single machine, namely the HDPA, proposed in this article, we are able to beat the state-of-the-art HCPA in terms of efficiency when the accuracy requirement is uniformly superior, e.g., above 0.99. In specifically, the HDPA combines VSSA probability updating functionality, discretizing the probability space and the Estimator phenomenon together into a hierarchical tree structuring with pursuit capabilities. We have explained the fine details of the algorithm and proven its ϵ-optimality through a formal, rigorous mathematical analysis. Our simulation results have demonstrated the advantage of the HDPA compared with the HCPA when the convergence criterion is close to unity. Indeed, the HDPA is arguably the best LA reported in the literature, to-date.

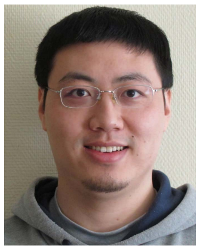
References

[1] M. L. Tsetlin, "Finite automata and modeling the simplest forms of behavior," *Uspekhi Matem Nauk*, vol. 8, no. 4, pp. 1–26, 1963.
[2] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction* (Dover Books on Electrical Engineering Series). Mineola, NY, USA: Dover, 2012.
[3] S. Lakshmivarahan, *Learning Algorithms Theory and Applications*, 1 ed. New York, NY, USA: Springer-Verlag, 1981.
[4] M. A. L. Thathachar and P. S. Sastry, "Estimator algorithms for learning automata," in *Proc. Platinum Jubilee Conf. Syst. Signal Process.* Bengaluru, Karnataka: Indian Institute of Science, Department of Electrical Engineering, 1986.
[5] R. Thapa, L. Jiao, B. J. Oommen, and A. Yazidi, "A learning automaton-based scheme for scheduling domestic shiftable loads in smart grids," *IEEE Access*, vol. 6, pp. 5348–5361, 2018.
[6] A. Yazidi, I. Hassan, H. L. Hammer, and B. J. Oommen, "Achieving fair load balancing by invoking a learning automata-based two-time-scale separation paradigm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3444–3457, Aug. 2021.
[7] S. Sahoo, B. Sahoo, and A. K. Turuk, "A learning automata-based scheduling for deadline sensitive task in the cloud," *IEEE Trans. Services Comput.*, vol. 14, no. 6, pp. 1662–1674, Nov. 2021.
[8] L. Zhu, K. Huang, Y. Hu, and X. Tai, "A self-adapting task scheduling algorithm for container cloud using learning automata," *IEEE Access*, vol. 9, pp. 81236–81252, 2021.
[9] R. R. Rout, G. Lingam, and D. V. L. N. Somayajulu, "Detection of malicious social bots using learning automata with URL features in Twitter network," *IEEE Trans. Computat. Social Syst.*, vol. 7, no. 4, pp. 1004–1018, Aug. 2020.
[10] H. Guo, S. Li, K. Qi, Y. Guo, and Z. Xu, "Learning automata based competition scheme to train deep neural networks," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 2, pp. 151–158, Apr. 2020.
[11] Z. Zhang, D. Wang, and J. Gao, "Learning automata-based multiagent reinforcement learning for optimization of cooperative tasks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4639–4652, Oct. 2020.
[12] C. Di, B. Zhang, Q. Liang, S. Li, and Y. Guo, "Learning automata-based access class barring scheme for massive random access in machine-to-machine communications," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6007–6017, Aug. 2019.
[13] S. Tanwar, S. Tyagi, N. Kumar, and M. S. Obaidat, "LA-MHR: Learning automata based multilevel heterogeneous routing for opportunistic shared spectrum access to enhance lifetime of WSN," *IEEE Syst. J.*, vol. 13, no. 1, pp. 313–323, Mar. 2019.
[14] B. El Khamlichi, D. H. N. Nguyen, J. El Abbadi, N. W. Rowe, and S. Kumar, "Learning automaton-based neighbor discovery for wireless networks using directional antennas," *IEEE Wireless Commun. Lett.*, vol. 8, no. 1, pp. 69–72, Feb. 2019.
[15] X. Deng et al., "Learning-automata-based confident information coverage barriers for smart ocean Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9919–9929, Oct. 2020.
[16] Z. Yang, Y. Liu, Y. Chen, and L. Jiao, "Learning automata based Q-learning for content placement in cooperative caching," *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3667–3680, Jun. 2020.
[17] R. O. Omslandseter, L. Jiao, Y. Liu, and B. J. Oommen, "User grouping and power allocation in NOMA systems: A novel semi-supervised reinforcement learning-based solution," *Pattern Anal. Appl.*, pp. 1–12, Jul. 2022, doi: 10.1007/s10044-022-01091-2.
[18] O.-C. Granmo, "The Tsetlin machine—A game theoretic bandit driven approach to optimal pattern recognition with propositional logic," 2018, *arXiv:1804.01508*.
[19] R. K. Yadav, L. Jiao, O.-C. Granmo, and M. Goodwin, "Robust interpretable text classification against spurious correlations using AND-rules with negation," in *Proc. 31st Int. Joint Conf. Artif. Intell.*, Jul. 2022, pp. 4439–4446.
[20] K. D. Abeyrathna et al., "Massively parallel and asynchronous tsetlin machine architecture supporting almost constant-time scaling," in *Proc. 38th Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 1–11.
[21] L. Jiao, X. Zhang, O.-C. Granmo, and K. D. Abeyrathna, "On the convergence of Tsetlin machines for the XOR operator," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Sep. 7, 2022, doi: 10.1109/TPAMI.2022.3203150.
[22] X. Zhang, L. Jiao, O.-C. Granmo, and M. Goodwin, "On the convergence of Tsetlin machines for the identity- and not operators," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6345–6359, Oct. 2022.
[23] S. Lakshmivarahan and M. A. L. Thathachar, "Absolutely expedient learning algorithms for stochastic automata," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, no. 3, pp. 281–286, Apr. 1973.
[24] B. Johnoommen, "Absorbing and ergodic discretized two-action learning automata," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-16, no. 2, pp. 282–293, Mar. 1986.
[25] B. J. Oommen and J. P. R. Christensen, "ϵ-optimal discretized linear reward-penalty learning automata," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-18, no. 3, pp. 451–458, Jun. 1988.

[26] X. Zhang, B. J. Oommen, and O.-C. Granmo, "The design of absorbing Bayesian pursuit algorithms and the formal analyses of their ϵ-optimality," *Pattern Anal. Appl.*, vol. 20, no. 3, pp. 797–808, Aug. 2017.
[27] M. Agache and B. J. Oommen, "Generalized pursuit learning schemes: New families of continuous and discretized learning automata," *IEEE Trans. Syst., Man, B, Cybern.*, vol. 32, no. 6, pp. 738–749, Dec. 2002.
[28] J. K. Lanctôt and B. J. Oommen, "Discretized estimator learning automata," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 22, no. 6, pp. 1473–1483, Nov./Dec. 1992.
[29] B. J. Oommen and M. Agache, "Continuous and discretized pursuit learning schemes: Various algorithms and their comparison," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 31, no. 3, pp. 277–287, Jun. 2001.
[30] X. Zhang, O.-C. Granmo, and B. J. Oommen, "Discretized Bayesian pursuit—A new scheme for reinforcement learning," in *Advanced Research in Applied Artificial Intelligence*, vol. 7345. Berlin, Germany: Springer, 2012, pp. 784–793.
[31] A. Yazidi, X. Zhang, L. Jiao, and B. J. Oommen, "The hierarchical continuous pursuit learning automation: A novel scheme for environments with large numbers of actions," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 2, pp. 512–526, Feb. 2020.
[32] G. I. Papadimitriou, "Hierarchical discretized pursuit nonlinear learning automata with rapid convergence and high accuracy," *IEEE Trans. Knowl. Data Eng.*, vol. 6, no. 4, pp. 654–659, Aug. 1994.
[33] X. Zhang, B. J. Oommen, O.-C. Granmo, and L. Jiao, "Using the theory of regular functions to formally prove the ϵ-optimality of discretized pursuit learning algorithms," in *Proc. 27th Int. Conf. Modern Adv. Appl. Intell.*, vol. 8481, 2014, pp. 379–388.
[34] K. Rajaraman and P. S. Sastry, "Finite time analysis of the pursuit algorithm for learning automata," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 26, no. 4, pp. 590–598, Aug. 1996.
[35] X. Zhang, L. Jiao, B. J. Oommen, and O.-C. Granmo, "A conclusive analysis of the finite-time behavior of the discretized pursuit learning automaton," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 1, pp. 284–294, Jan. 2020.
[36] X. Zhang, B. J. Oommen, O.-C. Granmo, and L. Jiao, "A formal proof of the ϵ-optimality of discretized pursuit algorithms," *Appl. Intell.*, vol. 44, no. 2, pp. 282–294, 2016.
[37] J. Zhang, C. Wang, D. Zang, and M. C. Zhou, "Incorporation of optimal computing budget allocation for ordinal optimization into learning automata," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 1008–1017, Apr. 2016.
[38] J. Zhang, C. Wang, and M. C. Zhou, "Fast and epsilon-optimal discretized pursuit learning automata," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2089–2099, Oct. 2015.
[39] J. Zhang, C. Wang, and M. C. Zhou, "Last-position elimination-based learning automata," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2484–2492, Dec. 2014.
[40] R. O. Omslandseter, "On the theory and applications of hierarchical learning automata and object migration automata," Ph.D. thesis, Dept. Inf. Commun. Technol., Univ. Agder, Norway, Europe, 2023.

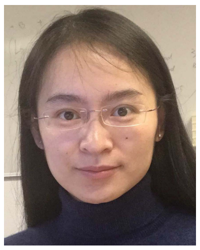
Rebekka Olsson Omslandseter was born in Porsgrunn, Norway, in January 1995. She received the B.E. degree from the University of Agder (UiA), Grimstad, Norway, in 2017, where she is currently pursuing the integrated Ph.D. degree.

She is currently a Scientific Researcher with the Centre of Artificial Intelligence Research (CAIR), UiA. Her research interests include learning automata, partitioning and clustering algorithms, resource allocation, and performance evaluation for communication and energy systems.



Lei Jiao (Senior Member, IEEE) received the B.E. degree from Hunan University, Changsha, China, in 2005, the M.E. degree from Shandong University, Jinan, China, in 2008, and the Ph.D. degree in information and communications technology from the University of Agder (UiA), Grimstad, Norway, in 2012.

He is currently an Associate Professor with the Department of Information and Communication Technology, UiA. His research interests include reinforcement learning, learning automata, natural language processing, and resource allocation for communication and energy systems.



Xuan Zhang received the bachelor's degree in electronics and information engineering and the master's degree in signal and information processing, and the Ph.D. degree in information and communication technology from the University of Agder (UiA), Grimstad, Norway, in 2005, 2008, and 2015, respectively.

She is currently a Senior Researcher with the Norwegian Research Center (NORCE), Grimstad. At the same time, she is also a Scientific Researcher with the Centre of Artificial Intelligence Research (CAIR), UiA. Her research interests include learning automata, mathematical analysis on learning algorithms, deep learning, natural language processing, and computer vision.

Dr. Zhang is currently serving as a Board Member of the Norwegian Association for Image Processing and Machine Learning.

Anis Yazidi (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees from the University of Agder, Grimstad, Norway, in 2008 and 2012, respectively.

He was a Researcher with Teknova AS, Grimstad. From 2014 to 2019, he was an Associate Professor with the Department of Computer Science, Oslo Metropolitan University, Oslo, Norway, where he is currently a Full Professor, leading the research group in applied artificial intelligence. He is also a Professor II with the Norwegian University of Science and Technology (NTNU), Trondheim, Norway. His current research interests include machine learning, learning automata, stochastic optimization, and autonomous computing.

B. John Oommen (Life Fellow, IEEE) was born in Coonoor, India, in September 1953. He received the B.Tech. degree from IIT Madras, Chennai, India, in 1975, the M.E. degree from the Indian Institute of Science, Bengaluru, India, in 1977, and the M.S. and Ph.D. degrees from Purdue University, West Lafayettte, IN, USA, in 1979 and 1982, respectively.

He joined the School of Computer Science, Carleton University, Ottawa, ON, Canada, in the 1981–1982 academic year. He is still at Carleton and holds the rank of a Full Professor. Since July 2006, he has been awarded the honorary rank of Chancellor's Professor, which is a Lifetime Award from Carleton University, and he is also an Extraordinary Professor at the North-West University, Potchefstroom, South Africa. His research interests include automata learning, adaptive data structures, statistical and syntactic pattern recognition, stochastic algorithms, and partitioning algorithms. He is the author of more than 495 refereed journal articles and conference publications.

Dr. Oommen is a fellow of IAPR. He has also served on the Editorial Board for the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS and *Pattern Recognition*.