Pioneering Approaches for Enhancing the Speed of Hierarchical LA by *Ordering* the Actions

Rebekka Olsson Omslandseter[a], Lei Jiao[a] and B. John Oommen[a,b]

[a]*Centre for Artificial Intelligence Research, University of Agder, Norway*
[b]*Carleton University, Ottawa, Canada*

ARTICLE INFO

Keywords:
Learning automata
Reinforcement learning
Hierarchical LA

ABSTRACT

Fixed Structure Stocastic Automata (FSSA), Variable Structure Learning Automata (VSSA), and their discretized versions have been significantly improved by utilizing inexpensive estimates of the actions' reward probabilities. These represent the fastest LA to date. However, the concept of *ordering* the actions has never been used within the field, and the reason for this is that there is no way to order the actions *a priori*. The recently-introduced Hierarchical Discrete Pursuit Automaton (HDPA) has an interesting concept of placing two-action LA along the nodes of a tree, implying that the leaves signify an underlying ordering. In this paper, we show that if estimates are available (as in the case of estimator algorithms), these can be used to place the actions at the leaf level to further enhance the convergence capabilities of the overall ensemble of the two-action LAs. This paper contains the design of this HDPA, the proof of this assertion, and the simulation results on benchmark Environments. Based on the results, we believe that it is the fastest and most accurate LA to date. Our position is that it will be very hard to beat its performance, since it has been incorporated *all* the salient features of the entire field of LA.

1. Introduction

The field of Learning Automata (LA)[1] was the forerunner for the entire domain of Reinforcement Learning (RL). The *Learning Automaton* is the Artificial Intelligence (AI) component that achieves the learning, and this is done through a sequence of feedback interactions between the LA and the entity called the *Environment*. The LA selects an action, either deterministically or stochastically. The Environment provides a feedback that is either a Reward or Penalty, and which is stochastic. This feedback is, typically, binary (as in this paper), but it can also be an element from a finite set, or a continuous range. The task of the designer is to specify the LA's learning policy by which it updates its knowledge based on the feedback. This record is maintained either in states that quantifies the memory, or in a so-called, action probability vector, as will be explained presently. The LA learns in a *semi-supervised* manner. This means that it does not require examples of solutions but learns via the feedback and in a trial-and-error mode. The metric for quantifying the performance of the LA is the average number of *iterations* it takes, over an ensemble of experiments, to attain a desired accuracy. For the Variable Structure Stochastic Automata (VSSA) type of LA used in this paper, the algorithm is set to have converged once it attains a specific probability level that is arbitrarily close to unity, and which is set by the user.

1.1. Memory Considerations

Learning cannot be achieved without remembering certain quantities during the process. We shall informally refer to these as the "*memory*". However, even if one remembers all the pertinent information in a very efficient manner, the learning will not succeed if the algorithm that utilizes the memory, is poor. To bring out the salient features of the pioneering contribution of this paper, we briefly itemize the respective components of the different families.

- **FSSA**: Fixed Structure Stochastic Automata (FSSA) are LA, where the memory is encapsulated in states which are identical to those possessed by Finite State Machines or flip flops. Examples of these are the Tsetlin, Krinsky, and Krylov LA [27]. In each case, the learning algorithm directs the LA to move across the states based on the

rebekka.o.omslandseter@uia.no (Rebekka Olsson Omslandseter); lei.jiao@uia.no (Lei Jiao); oommen@scs.carleton.ca (B. John Oommen)
ORCID(s): 0000-0002-8271-7325 (Rebekka Olsson Omslandseter); 0000-0002-7115-6489 (Lei Jiao); 0000-0002-5105-1575 (B. John Oommen)

[1]The term LA is used to address both the field of Learning Automata or the Learning Automata themselves, depending on the context.

response from the Environment, and each of the latter boast their own individual strategy. Correspondingly, they all have different convergence characteristics.

- **VSSA**: Unlike FSSA, in VSSA, the memory is contained in the action probability vector, $P(n)$. The action is chosen based on $P(n)$, which is then communicated to the Environment. $P(n+1)$ is obtained in the next step, and is based on $P(n)$, the action chosen, $\alpha(n)$, and the feedback that the Environment provides, $\beta(n)$. The updating algorithm, on the other hand, can be varied and includes, among others, the Linear Reward-Penalty (L_{R-P}) scheme, the Linear Reward-Inaction (L_{R-I}) scheme, the Linear Inaction-Penalty (L_{I-P}) scheme, and the Linear Reward-ϵPenalty ($L_{R-\epsilon P}$) [9], [11].
- **Estimator Algorithms**: In Estimator LA, the memory resides in the action probability vector and running estimates of the reward probabilities. In the Pursuit algorithm, the learning algorithm now increases the probability of the currently recorded best action and not of the action that is chosen. In the Pursuit algorithm, only the probability of the best action is increased. In the Generalized Pursuit, the action probabilities of a subset of actions are increased, while the rest of the probabilities are decreased. One has to mention that the estimates can be done in an ML or Bayesian manner [33], and the updates done in a continuous or a discretized paradigm.
- **Hierarchical LA:** In the family of hierarchical LA, the machines are arranged in a tree structure, and the actions are at the leaves. These individual LA can be VSSA or can be Pursuit machines themselves. More details of this are included in the next section.

The above bullets briefly encapsulates the entire prior art.

1.2. Action Ordering Considerations

The reader will observe that throughout the above discussions, the ordering of the actions has remained insignificant. This is, of course, valid because the ordering is unknown unless one resorts to a prior Random Race competition which is not relevant to our present study [12]. Of course, the ordering of the actions in an action probability vector is also meaningless.

The hypothesis of this study is that there is an advantage to ordering the actions. Clearly, such an ordering can only be enforced if there are crude estimates of the reward probabilities. If they are arranged linearly, ordering the actions can enhance the corresponding choice by resorting to a fast searching mechanism, as opposed to a linear search. We shall not elaborate on that issue here.

However, let us consider the case when the automata operate in a hierarchical manner. The actions then are placed at the leaves of the tree, and the decisions of the individual LA trickle up to the root. Our hypothesis is that rather than keeping the leaves completely unordered, information gleaned during the initial learning phase can be used to order them, and to yield a superior performance. This is precisely the hypothesis and contribution of our paper.

1.3. Contributions of this Paper

The contributions of this paper can be summarized as follows:

- Unlike the prior art, we show that there is an advantage in considering the ordering of the actions when the LA operate in a hierarchical manner.
- We demonstrate this, by considering the most recent machine in the field, the HDPA.
- We confirm the hypothesis, by reporting the results of extensive simulations in different Environments and a host of distributions.

As mentioned above, the concept presented in this paper is true, not only for the HDPA, but also for any other type of machine utilizing a hierarchical structure.

2. Related Work

The paradigm of LA originated in the 1960s with Michael Lvovitch Tsetlin and his innovation of learning agents and, ultimately, the Tsetlin Automata [26]. Later advancements followed, and the types of LA are generalized into two categories, namely FSSA and the VSSA. Both in the FSSA and the VSSA, the LA learns over time, selecting with

increasing probability the action that ensures it the highest likelihood of being rewarded by the Environment. FSSA has a state-based structure, where the current state of the LA determines its action. Consequently, the automaton learns through interactions with the Environment, traversing the states according to a fixed or stochastic learning scheme. With VSSA, the LA determines its action by randomly sampling its action probability vector. Thus, the automaton maintains a vector with probabilities of choosing the different actions it can choose from and updates the probabilities following its interactions with the Environment. The discovery of VSSA was a quantum step in terms of LA's efficiency [11].

2.1. Discretizing the Probability Space

In VSSA, we have the Linear Reward-Penalty (L_{R-P}) scheme, the Linear Reward-Inaction (L_{R-I}) scheme, the Linear Inaction-Penalty (L_{I-P}) scheme, and the Linear Reward-ϵPenalty ($L_{R-\epsilon P}$) [9], [11]. In these different schemes, the probability vector is updated linearly. The updating can also be done in a non-linear manner [9], [11], [8]. At the same time, VSSA schemes can be continuous or discrete [19]. Continuous type VSSA updates the probability in a multiplicative manner with a factor, while the discretized type updates the probability with a constant in each update. Due to the multiplicative updating, the continuous type can experience slower algorithm speeds than the discretized type. Thus, when an action selection probability is closer and closer to unity, the change in its probability becomes less and less. The continuous and discrete updating functionality has been investigated mathematically in [20], [32].

2.2. Introducing the Concept of Estimation

Another major discovery in the field of LA was the Estimator-based Algorithms (EAs), which significantly increased the convergence speed of VSSA [25]. The concept of EAs is the utilization of estimation. In more detail, the LA keeps reward estimates while in operation, using these estimates to pursue the currently most promising action (referred to as *Pursuit* in the Literature) [34]. Researchers combined the Pursuit concept with discretized updating, leading to the paradigm of Discrete Estimator Algorithms (DEAs) [10], superior to earlier VSSA variants in terms of convergence speed.

2.3. Incorporating Structure

Although all of the advances mentioned above increased the applicability and efficiency of LA dramatically, VSSA still had an impediment as the number of actions (possible solutions to a problem) became *large* (e.g., more than ten [30]). The action selection probability vector in VSSA sums to unity. If there are R actions, and R is large, the action selection probabilities of the actions will become so small that they might not even be chosen. The principle of VSSA, and its ability to find the optimal action, depends on all actions being sufficiently explored. This challenge makes it harder for the LA to find the optimal action, enforcing utilization of a smaller learning parameter, leading the algorithm to become less efficient (requiring more iterations before convergence). Therefore, the authors of [30] proposed the HCPA, bringing structure to the domain of VSSA. The HCPA was a quantum step to the field of LA, making VSSA able to handle a large number of actions. However, as the accuracy requirement to the HCPA became large, e.g., above 0.98, the HCPA suffered from its multiplicative property of updating its action selection probabilities, resulting in sluggish convergence. In [14], the HDPA was proposed, combining VSSA, discretized updating, the Pursuit concept, and structure. The HDPA provides a solution to problems with many actions and high accuracy requirements, constituting the state-of-the-art for generic LA, being significantly faster when compared to the HCPA. For both the HCPA and HDPA, all the LA in the tree were configured to have two actions. Although the LA in the tree can have more than two actions, their design is targeted at mitigating the diminishing increase/decrease of the action selection probabilities as R becomes large. Therefore, the number of actions per LA should be kept low, and in all the papers, it has been set to two.

2.4. Applications

LA has a wide range of applications. To demonstrate the variety of applications for LA, we mention here that LA has been utilized for scheduling shiftable loads in Smart Grids [24], for obtaining fair load balancing in cloud based services [31] (and cloud task scheduling [22, 37]), for finding malicious bots in social networks [21], and training deep neural networks [5]. LA has also been arranged to solve cooperative tasks [36], and for solving the massive access problems in Machine-to-Machine communications [2]. Furthermore, LA has been utilized in wireless sensor networks for energy conservation [23], neighbor discovery [7], and establishing barrier paths [3]. In mobile radio communication, LA was used in a cooperative caching scheme [29], and in a system designed for 6G and beyond, LA was used for user grouping [18]. Additionally, LA has been implemented into more sophisticated learning algorithms like, i.e., the Tsetlin

Machine (TM) [4]. The TM is interpretable and produces competitive results in comparison to neural networks, and has demonstrated promising results in text classification [28]. A massively parallell and asyncronous TM was introduced in [1], and its convergence behavior was analyzed in [6, 35].

2.5. The Differences between Deep Learning and LA

Deep Learning (DL) and LA are two distinct paradigms within machine learning with different principles, advantages, and applications. LA is a probabilistic learning approach, and the algorithms within this field generally make decisions based on the probability distribution of the possible actions (solutions) within a random environment [13]. More specifically, the LA gradually refines its strategy based on the concept of reinforcement learning, i.e., rewards and penalties for a wanted or unwanted behavior. LA is suitable for scenarios with structured and deterministic responses and requires relatively small computational resources compared with DL. One of the most important behaviors of LA is its ability to tackle highly stochastic environments [14]. However, LA can suffer from limitations in complex, high-dimensional scenarios due to its reliance on explicit reward/penalty feedback. Conversely, DL relies on neural networks with many layers, which automatically learn hierarchical representations from raw data [15]. DL is a good fit for handling unstructured and high-dimensional data such as images, text, and sound, where backpropagation and gradient descent is involved, in an iterative manner, to adjust network weights to minimize the loss [16]. However, DL models often require substantial volumes of labeled data, which is a requirement that increases with the number of dimensions in the dataset, often accompanied by a significant requirement to computational resources for training and inference [15, 17]. Hence, while both methods learn from experiences and certain feedback, the intrinsic difference lies in their learning strategies, the nature of data they can efficiently process, and their resource requirements.

3. Incorporating Ordering into a Hierarchical LA

Although the HDPA improved the convergence speed significantly for high accuracy requirements, we experienced that the convergence speeds were substantially dependent on the action distribution on the leaf level of the tree. Linked to the concept of Random Races [12], where the LA is modeled to find an ordering of the actions in an ascending/descending order, we hypothesize that we can order the actions in a manner that is beneficial to the algorithm. More specifically, we propose that we can use an Estimation Phase in the HDPA process and also the estimated reward probabilities to reorder the actions at the leaf level to yield an improved performance. Consequently, in this paper, we propose the Action Distribution Enhancing (ADE) approach for enhancing the convergence speed of the HCPA and HDPA. The improvement in the convergence speed becomes more noticeable as the number of actions at the leaf level increases. Therefore, organizing the actions in an improved manner can significantly reduce the number of iterations before the convergence is achieved. While this was our intended hypothesis, as demonstrated through extensive simulations documented later in the paper, we show that the ADE approach is, indeed, beneficial compared to randomly initializing the actions at the leaf level of the tree. The reader should note that the problems that LA can solve are random in nature, and for a real problem, no information about the reward probabilities can be known *a priori*. Therefore, understandably, the Estimation Phase is needed to obtain an improved ordering.

3.1. Motivating arguments

To motivate the development of our new paradigm, we consider a problem involving four actions $\mathcal{A} = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ with the corresponding estimated action probability vector $\hat{D} = \{\hat{d}_1, \hat{d}_2, \hat{d}_3, \hat{d}_4\}$, taken over an initial estimation phase of 20 iterations. The reader must please observe that because the number of iterations are small, the corresponding estimates will be inaccurate. Also, before we proceed with the arguments, it is wise to see how these estimates will effect the learning process. Further, in the interest of simplicity, we proceed with the discussion by considering the case of the HDPA instead of any other arbitrary hierarchical LA.

At the leaf levels, the four actions are to be placed in one of the 4! positions. It is also obvious that the descending and ascending orders of the placements of the actions are merely mirror reflections of each other. On a deeper examination of a Hierarchical Pursuit LA, one observes that from every level, the estimates of the most suitable actions chosen at that level will be trickled up. This implies that the automata at each level will be dealing with problems of different complexities. However, the most important automaton is the one placed at the root, because that governs, or rather dictates, the operations of all the automata below it.

Consider the following figure in which the four actions are placed at the leaves in the descending order (Fig. 1a). From Fig. 1a, we see that the LA $A_{1,1}$ has to deal with actions whose reward estimates are $\frac{15}{20}$ and $\frac{11}{20}$. Similarly, when

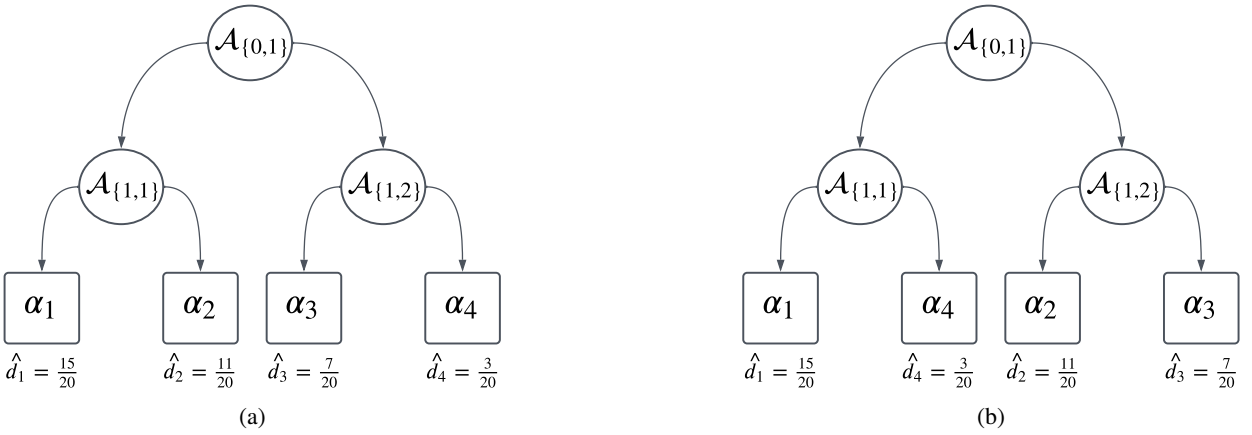

Figure 1: Example of two hierarchical tree structures for four actions with different action distributions at the leaf level.

the actions are in a more random order, Fig. 1b, the corresponding reward estimates for the LA $A_{1,1}$ are $\frac{15}{20}$ and $\frac{3}{20}$. If all goes well as in a perfect world, the trickled up estimates are those of the optimal actions of their corresponding children. The root level, which is encountering the most important task, has now to deal with distinguishing between the reward estimates $\frac{15}{20}$ and $\frac{7}{20}$ in Fig. 1a, and $\frac{15}{20}$ and $\frac{11}{20}$ in Fig. 1b. This means that the root automaton, that has to solve the most discriminating problem of all the automata, has to resolve actions α_1 and α_2, in the case of Fig. 1b, which is much more difficult than the problem in Fig. 1a. This constitutes the rationale for our strategy and is formalized below.

When we use the euphemistic expression above "if all goes well in a perfect world", we emphasize that it is statistically not at all unrealistic. This is because by the law of large numbers or the Estimation Phase in the Bayesian case, the estimates of the reward probabilities will converge to their true values with an arbitrarily high accuracy. Thus, the asymptotic arguments (and probabilities) of the trees of Fig. 1a and Fig. 1b will still be valid.

3.2. Proof

While the above argument has merely been shown by a verbal explanation, the formal mathematical analysis is straightforward, and obtained by invoking simple min and max arguments. It is given below.

Theorem 1. *A Hierarchical Pursuit LA, in which the best action and the second best action are in different sub-trees of the root node, is always inferior to the same Hierarchical LA where they fall in the same sub-tree.*

PROOF OF THEOREM 1. We preface the proof by asserting that a more complex problem at the root level, will require more iterations. We now consider the two scenarios which are representative of the corresponding competing problems.

Consider the Hierarchical LA. Let us assume that α_1 and α_2 are the best and the second best actions, and that their reward estimates are $\hat{d}_1$ and $\hat{d}_2$, respectively. We now consider two mutually exclusive and collectively exhaustive cases. The first, Case 1, is the one in which α_1 and α_2 are in different sub-trees, for example, the left and right sub-tree. Of course, the mirror case where α_2 is in the left sub-tree and α_1 is in the right, are identical. In other words, α_1 and α_2 are in different sub-trees as viewed from the root. Thus, the learning problem concerns differentiating between α_1 and the rest of the actions in its sub-tree, and α_2 and the rest of the actions in its corresponding sub-tree. By a simple inductive argument, we see that at the first level, the competing actions become α_1 and α_2, which is, of course the most difficult discriminating problem.

As opposed to this, in the tree of Fig. 1a, the most superior action in the left sub-tree becomes α_1, and it has now to compete with some action α_j where $j \neq 2$. This is obviously, a less complex problem than the one encountered in Fig. 1b, implying that, the scenario in Fig. 1a will lead to a faster convergence.

All of these arguments, indeed, follow by virtue of the properties of the min and max operators, and the unnecessary details are omitted.

4. The Action Distribution Enhancing (ADE) Approach

As mentioned earlier, such an ADE approach applies to both the HCPA and the HDPA, and indeed, to any hierarchical LA. However, because the HDPA has demonstrated superior performance to the HCPA for high accuracy requirements, and we have limited space, we only highlight the ADE approach for the HDPA, by understanding that the principles for the HCPA are analogous. The ADE approach for the HCPA is similar to the approach explained for the HDPA.

The ADE approach concerns distributing the actions at the leaf level of the hierarchical tree in an improved manner. For a practical, real-life problem, there can be little information as *a priori* information about the actions. This is why the distribution of the actions at the leaf level is an intricate problem which has not yet been considered in the Literature. The ADE approach is two-pronged. The first prong is the *Estimation Phase*, used for estimating the action reward probabilities. The second concerns distributing the actions in an improved manner, and is referred to as the *Reallocation Process*.

The first part of the ADE approach concerns the Estimation Phase. In the HDPA, the estimated reward probability and the action selection probabilities of LAs throughout the tree structure are initialized as 0.5, according to [30] and [14]. Thus, the HDPA estimates the reward probabilities as per the Pursuit concept (maintaining an estimated reward probability vector). We propose a standalone Estimation Phase with the ADE approach prior to the HDPA starting its regular operation. This means that in this phase, we include θ iterations per action for estimating their reward probabilities. These estimates are further utilized as to initialize the corresponding values in the regular operation of the HDPA, which happens after the Reallocation Process.

The second part of the ADE approach concerns the Reallocation Process, which distributes the actions in an improved manner. For a two-action LA configured tree, as proven in Theorem 1, such an organization can be achieved by ordering the actions according to their estimated reward probabilities in either an ascending or descending order. In this way, asymptotically, the optimal and second optimal actions will share the same LA at the level below the root. Thus, asymptotically, the algorithm will have achieved a correct estimation of the reward probabilities, and the actions will be distributed in an improved manner. Clearly, due to the stochastic nature of the problem, the Estimation Phase might not always yield a perfect interpretation of the reward estimates and the corresponding trees. This phenomenon and its consequences are explained further in the section with the experimental results (Section 5).

The Reallocation Process uses the estimated reward probabilities from the Estimation Phase to reallocate the actions to the tree's leaves in an ascending/descending order. Thus, after the Estimation Phase, the actions are given a new location at the leaf level. The reader should note that the estimates also need to be updated according to this new ordering. By maintaining these estimates, the information from the Estimation Phase is also retained. After the Reallocation Process, the HDPA starts its *normal* operation, by utilizing the reward estimates from the first phase.

The concept of the HDPA and the hierarchical structure in LA can be linked with human behavior. One well-known attribute of human behavior is sequential decision-making. This refers to the case of making a series of decisions, where each decision (action) is related to the outcome of the previous decisions (actions). Similarly, in a hierarchical tree structure, as in the HDPA, the actions are ordered sequentially, starting from the root node, proceeding down the tree to its child nodes, by following different paths based on the conditions at each tree level. Each decision (action) leads to a new set of possible actions, which reflects the evaluative nature of human decision-making.

The ADE approach can also be linked with human behavior. The ADE approach is based on the idea that the order of the leaves - specifically, the prioritization of the actions' estimated reward probability- can significantly influence the algorithm's efficiency. This process mirrors the way humans often approach different and complex problems. Whether consciously or subconsciously, humans evaluate and rank choices based on their perceived potential for success. When faced with multiple possible solutions, we prioritize them based on specific success metrics or "fitness" and start with the most promising ones. Thus, the ADE approach reflects the human nature of problem-solving behavior by prioritizing options based on their likelihood of success, i.e., their estimated reward probability. The ordering at the leaf level makes it easier for the automaton to explore the actions optimally. Concentrating the exploration of choices within a particular part of the tree is similar to how humans often compartmentalize decisions. For instance, when planning a trip, we first focus on choosing the trip's destination before considering associated decisions like accommodation and activities. In this way, the most promising or *important* choices are considered first, simplifying the overall decision-making process.

Since a detailed explanation of the HCPA and HDPA can be found in [30] and [14], respectively, we concentrate on the ADE phase and the creation of the corresponding tree. The algorithm is formally presented in Algorithm 1, and

the notations are similar to those established in these papers. The reader should note that the description submitted below does not include the updating of the estimated reward probabilities along the path. This is to avoid it from being unnecessarily cumbersome. As mentioned earlier, if we want to use the HCPA (or any other type of Hierarchical LA) instead, we only need to change the reference to the HDPA to that of the corresponding variant.

Algorithm 1 The ADE HDPA

Required Parameters:

K: The number of levels in the tree, where k is used to index the depth in the tree and $k \in \{0, 1, ..., K\}$.
$R = 2^K$: The number of actions.
$\alpha_{\{K,j\}}$: Action at the leaf level of the tree, where $j \in \{1, 2, ..., R\}$.
T: The convergence criterion threshold.
t: The number of iterations.
Δ: The learning parameter, where $0 < \Delta < 1$.
θ: The number of iterations used per action in the Estimation Phase.
β: Environment's response, where $\beta = 0$ is a Reward and $\beta = 1$ is a Penalty.
$u_{\{K,j\}}$: The number of times that action $\alpha_{\{K,j\}}$ was rewarded ($j \in \{1, 2, ..., 2^K\}$).
$v_{\{K,j\}}$: The number of times that action $\alpha_{\{K,j\}}$ was selected ($j \in \{1, 2, ..., 2^K\}$).
$\hat{d}_{\{K,j\}} = \frac{u_{\{K,j\}}}{v_{\{K,j\}}}$: The estimated reward probability of action $\alpha_{\{K,j\}}$, where $j \in \{1, 2, ..., 2^K\}$, and the vector is initialized as 0.5, i.e., $\hat{d}_{\{K,j\}}(0) = \frac{1}{2}$.
$\mathcal{A}_{\{k,j\}}$: Notation of the LA in the tree, where for each level k, we have $j \in \{1, 2, .., 2^k\}$.
$\mathcal{P}_{\{k,j\}}$: Each LA have an action probability vector $\mathcal{P}_{\{k,j\}} = [p_{\{k+1,2j-1\}}, p_{\{k+1,2j\}}]$.

Algorithm Begin

// The Estimation Phase
$r = 0$
for $R \in \{1, 2, ..., R\}$ **do**
  $r = r + 1$
  for θ **do**
    $v_{\{K,r\}} = v_{\{K,r\}} + 1$
    – Test $\alpha_{K,r}$ with the Environment.
    if $\beta = 0$ **then** *// The Environment responds with a Reward.*
      $u_{\{K,r\}} = u_{\{K,r\}} + 1$
    end if
  end for
end for
$\hat{d}_{\{K,j\}} = \frac{u_{\{K,j\}}}{v_{\{K,j\}}}, \forall j, j \in \{1, 2, ..., 2^K\}$ *// Update the estimated reward probability.*
// The Reallocation Process

- Based on $\hat{d}_{\{K,j\}}$, reallocate the actions at the leaf level in asc/desc order.
- Based on $\hat{d}_{\{K,j\}}$, reallocate the estimated reward probability in asc/desc order.

$t = \theta R$
// The Normal Operation
while the convergence criterion is not met **do**
  – Continue the HDPA operation as explained in Algorithm 2.
end while *// The Algorithm have Converged.*

Algorithm End

5. Experimental Results

To demonstrate the effectiveness of the proposed ADE approach presented in this paper, we conducted experiments with various action distributions, numbers of actions, and Environments. For quantifying the effectiveness of the LA algorithms, we recorded the number of iterations required before convergence, as this is the most common evaluating measurement [11]. As mentioned earlier, the convergence requirement for VSSA is that one of the action selection probabilities attains a certain threshold (T). In our experiments, we tested different convergence criteria, and report the results for the most utilized convergence threshold to be 0.995. The reader should note that this metric is helpful

Algorithm 2 The HDPA

1. $t = t + 1$
2. **For levels** $k \in \{0, ..., K-1\}$:
   - The LA $\mathcal{A}_{\{k,j\}}$ selects an action by randomly sampling from an uniform random distribution in accordance with its action selection probabilities in $\mathcal{P}_{k,j}$.
   - We denote the action chosen as $j_{\{k+1\}}(t)$, where $j_{\{k+1\}}(t) \in \{2j-1, 2j\}$.
   - The action chosen activates the LA at the level below as $\mathcal{A}_{\{k+1,j_{\{k+1\}}(t)\}}$.
   - The outlined process above continues including $k = K-1$.
3. **For level** $k = K$:
   - We denote the action that was selected by the LA at level $K-1$ as $j_K(t)$, where $j_K(t) \in \{1, \ldots, 2^K\}$.
   - Based on the response from the Environment, we update $\hat{d}_{\{K,j_K(t)\}}(t)$ as:
     $u_{\{K,j_K(t)\}}(t+1) = u_{\{K,j_K(t)\}}(t) + (1-\beta(t))$
     $v_{\{K,j_K(t)\}}(t+1) = v_{\{K,j_K(t)\}}(t) + 1$
   - All the other leaves, where $j \in \{1, ..., 2^K\}$ and $j \neq j_K(t)$ are kept as:
     $u_{\{K,j\}}(t+1) = u_{\{K,j\}}(t)$
     $v_{\{K,j\}}(t+1) = v_{\{K,j\}}(t)$
   - And the overall $\hat{d}_{\{K,j\}}(t+1)$ is updated as:
     $\hat{d}_{\{K,j\}}(t+1) = \frac{u_{\{K,j\}}(t+1)}{v_{\{K,j\}}(t+1)}$.
4. Next, we update the action selection probabilities to the currently best estimated action and along its reverse path. The reader should note that the best estimated action is not necessarily the action that was chosen in this iteration. We denote $h_K(t)$ as the index of the currently best estimated action at the leaf level. The parent of $h_K(t)$ can be found as $\mathcal{A}_{\{K-1,\lceil h_K(t)/2 \rceil\}}$, and we refer to its other child as $\overline{h_K(t)} = \{2h_K(t)-1, 2h_K(t)\} \setminus h_K(t)$.
   - First, the best estimated action is rewarded at $h_K(t)$'s parent:
     if $\beta(t) = 0$ **then**
     $p_{\{K,h_K(t)\}}(t+1) = \min\left(p_{\{K,h_K(t)\}}(t) + \Delta, 1\right)$
     $p_{\{K,\overline{h_K(t)}\}}(t+1) = 1 - p_{\{K,h_K(t)\}}(t+1)$.
     end if
   - After that, the LA along the reverse path become recursively updated, including the LA at the top:
     for $k \in \{K-1, ..., 1\}$
     We let the action on the path to the estimated best action be $h_k(t) = \lceil h_{k+1}/2 \rceil$, and $\overline{h_k(t)}$ be the index of the opposite action, i.e., the other action in the same LA as $h_k(t)$.
     if $\beta(t) = 0$ **then**
     $p_{\{k,h_k(t)\}}(t+1) = \min\left(p_{\{k,h_k(t)\}}(t) + \Delta, 1\right)$
     $p_{\{k,\overline{h_k(t)}\}}(t+1) = 1 - p_{\{k,h_k(t)\}}(t+1)$.
     end if
     end for

because it will remain identical, regardless of the computing power on the machine used for the experiments, or the efficiency of the code or language used[2].

Paired together with this, is the accuracy of the algorithm. Due to the stochastic nature of the problem, one often conducts more experiments and reports the average performance. In terms of accuracy, we usually measure this as the percentage of experiments which have converged to the optimal action. Consequently, when a 100% accuracy is

[2]In our experiments, we have configured the convergence criterion as being achieved once *any* of the LA has attained a certain threshold of choosing one of the actions in its action probability vector. However, in [14], they defined the convergence as being achieved only when all the LA along the path to a leaf action had attained the prescribed threshold. Thus, the convergence criterion in this paper is different, i.e., it utilizes the "logical or" instead of the "logical and", making the algorithms (i.e., both the HDPA without/with the ADE) attain a faster convergence.

Table 1
Experimental results for different action distributions without the ADE approach for eight actions with $T = 0.995$ as the convergence criterion. The results were averaged over $1,000$ experiments, with $\Delta = 9e{-}5$.

Config.	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	Avg	Std	Acc.
1	0.99	0.95	0.87	0.6	0.43	0.54	0.67	0.51	6,727.40	42.76	100%
2	0.87	0.6	0.43	0.54	0.67	0.51	0.99	0.95	6,791.75	56.81	100%
3	0.87	0.6	0.99	0.95	0.43	0.54	0.67	0.51	6,730.42	42.03	100%
4	0.87	0.6	0.43	0.99	0.95	0.54	0.67	0.51	7,082.38	215.46	100%
5	0.99	0.6	0.43	0.54	0.67	0.51	0.87	0.95	7,109.98	220.78	100%
6	0.99	0.99	0.51	0.67	0.54	0.43	0.6	0.87	6,791.15	57.52	100%
7	0.99	0.99	0.87	0.67	0.6	0.54	0.51	0.43	6,713.85	42.49	100%

achieved, all the experiments conducted in an ensemble of experiments have converged to the optimal action (i.e, the action with the highest reward probability). The reader should note that the number of iterations used for the Estimation Phase is also reckoned into the overall number of iterations in our experimental results.

In LA, the tuning of the learning parameter leads to a trade-off between the accuracy and the speed. Generally, as the learning parameter becomes smaller, the number of iterations increases, and at the same time, the algorithm performs more accurately. The same dilemma applies to the algorithm proposed in this paper. As demonstrated in the experiments, placing the optimal and sub-optimal actions in opposite parts of the tree at the leaf level requires more iterations than establishing them as entities in the same part of the tree (for example, i.e., with an ascending/descending ordering), concurring with Therorem 1. Thus, the ascending/descending orders generally require substantially less number of iterations before convergence. Therefore, it might be harder for the algorithm to attain to an accurate convergence with the same learning parameter used when the optimal and sub-optimal are in opposite parts of the tree. Because the ascending/descending orders are considerably faster, it might require a smaller learning parameter to converge accurately.

With our experiments, we wanted to demonstrate the behavior of the HDPA for different action distributions at the leaf level, thereby demonstrating the improved performance with the ADE approach. In practice, we have little or no *a priori* information about the reward estimates in a real-life scenario. In our simulations, we know that with a knowledge of the exact reward probabilities, we are able to execute the programs for different action distributions with and without the ADE approach. By demonstrating the phenomena for different configurations, we intend to highlight the improved performance that can be achieved in a real-life scenario by using the ADE approach with the Estimation Phase and Reallocation Process before the actual LA learning is performed.

In our simulations, we denote the *real* reward probabilities, i.e., the probability of getting a reward for selecting a certain action, as d_j, where $j \in \{1, 2, ..., 2^K\}$. If nothing else is specified, for the ADE HDPA, we used a descending ordering in the Reallocation Process. To help understand the ordering at the leaf level, we present visualizations of the action distributions with their corresponding reward probabilities. Please note that these visualizations show the real reward probability of α_1 to α_{2^K}, i.e., d_1 to d_{2^K}, for the actions at the leaf level. Consequently, the HDPA without the ADE will run the experiments with the actions ordered as displayed by the configuration. Conversely, the ADE HDPA will reallocate the actions at the leaf level in an ascending/descending order based on the corresponding reward estimates.

5.1. Simulation 1: 8 Actions

Let us first consider the results for Simulation 1 presented in Tables 1 and 2, which involve Environments of 8 actions. The difference between the two tables is that Table 1 does not incorporate the ADE, and Table 2 does. The categoric superiority of the results in Table 2 demonstrates the power of the ADE.

In Simulation 1, we ran 1,000 experiments with similar reward probabilities associated with the Environment, but with different action distributions. In Fig. 2, we visualize some of the different action distributions encountered in Simulation 1. For example, we can observe Config. 5, where the optimal and sub-optimal actions are located in opposite parts of the tree. We had earlier proven that this configuration was the hardest for the HDPA to handle. As opposed to this, having the actions with regard to the reward probabilities in an ascending/descending order, like the case of Config. 7, would be easier. This is confirmed from Table 1, where the HDPA with the ADE has a superior performance, demonstrating the assertion of Theorem 1. Config. 4 and Config. 5 required the highest number of iterations and had

Table 2
Experimental results for different action distributions with the ADE HDPA with eight actions and $T = 0.995$ as the convergence criterion. The results were averaged over $1,000$ experiments, with $\Delta = 9e{-}5$.

Config.	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	Avg	Std	Acc.
1	0.99	0.95	0.87	0.6	0.43	0.54	0.67	0.51	6,810.10	44.66	100%
2	0.87	0.6	0.43	0.54	0.67	0.51	0.99	0.95	6,824.48	51.26	100%
3	0.87	0.6	0.99	0.95	0.43	0.54	0.67	0.51	6,823.52	49.19	100%
4	0.87	0.6	0.43	0.99	0.95	0.54	0.67	0.51	6,821.79	49.51	100%
5	0.99	0.6	0.43	0.54	0.67	0.51	0.87	0.95	6,825.32	49.49	100%
6	0.95	0.99	0.51	0.67	0.54	0.43	0.6	0.87	6,813.32	47.15	100%
7	0.99	0.95	0.87	0.67	0.6	0.54	0.51	0.43	6,810.91	44.67	100%

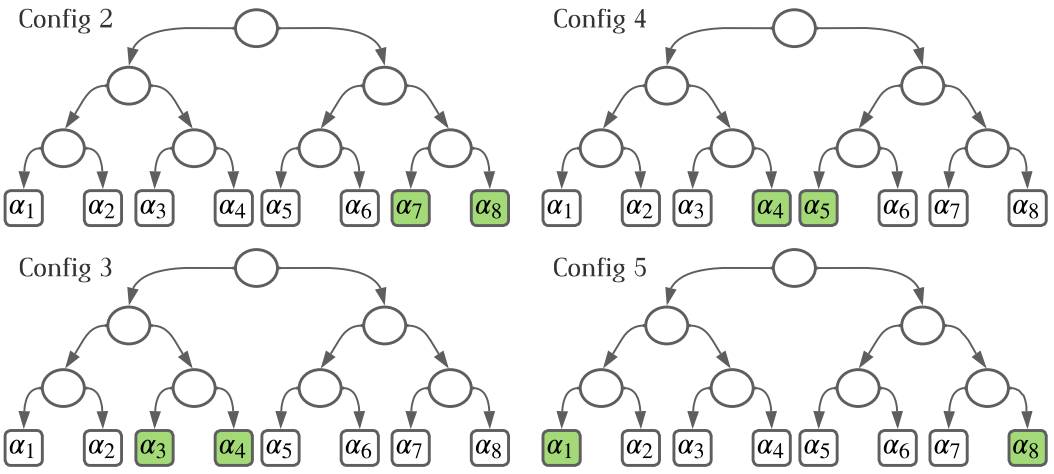

Figure 2: The figure shows the tree structure for four different action distributions in Simulation 1, where the optimal and the second optimal actions are marked with different backgrounds at the leaf level.

high standard deviations (Std). However, for Config. 7, where the actions were ordered in a descending order according to the reward probabilities, the number of iterations was the lowest. This is obvious since this is the most optimal setting. Again, the results for the ascending and descending orders are almost identical.

In Table 2 we present the results for the experiments for the ADE HDPA for Simulation 1. As we observe from the table, some configurations required more iterations than the other configurations without the ADE approach. However, the ADE HDPA was more consistent in the number of iterations, and had a more stable and smaller standard deviation. Config. 2, which is the reversed form of Config. 6, yielded a little higher standard deviation than the others. We also, tested Config. 2 for the ADE HDPA with an ascending ordering in the Reallocation Process, and these experimental results were similar to those of Config. 6. More specifically, for 1,000 experiments with the ADE HDPA and an ascending ordering in its Reallocation Process, the algorithm required 6,818.14 iterations on average and a standard deviation of 48.46 yielded 100% accuracy. The corresponding figure without the ADE was iterations, which is xx percent higher.

In real-life, we do not know the underlying reward probabilities. Therefore, we can only tune the number of tests, θ, that is used in the Estimation Phase. Testing each action for a larger number of iterations in the Estimation Phase will make the ADE more certain that it has estimated the reward probabilities correctly, and it will, thus, order them correctly as well. However, in most cases, because we want a fast convergence, a rough estimate might be sufficient. In Simulation 1, we used $\theta = 12$. If we had *perfect* estimation of the reward probabilities, we would have similar results to Config. 7 without the ADE.

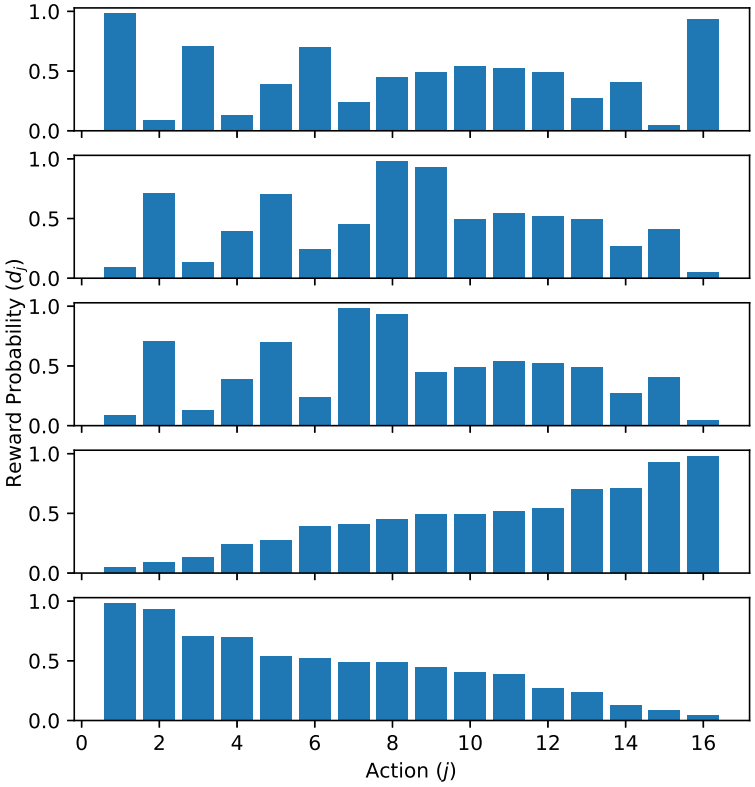

Figure 3: The figure shows the action distribution in accordance with the reward probabilities for Simulation 2. Config. 1 is at the top, Config. 5 is at the bottom, and the others are ordered in between them systematically.

5.2. Simulation 2: 16 Actions

In Simulation 2, we increased the number of actions to 16. Again, we set the value of θ to be 12. The different original action distributions are visualized in Fig. 3, where we focus on the optimal and sub-optimal actions' locations in the different configurations. However, the actions in between them were distributed more or less *randomly*. In subsequent simulations, we shall highlight what happens when we have more constructed forms in the original distributions.

In Tables 3 and 4, we present the results for Simulation 2. Analogous to Simulation 1, we see that the number of iterations is more consistent, and that the standard deviation is smaller for the ADE HDPA (Table 4). The HDPA without ADE still had better results for the configurations where the actions were manually ordered in an ascending/descending order, which is quite understandable. Thus, the ADE HDPA did probably not achieve the *perfect* estimation of the reward probabilities, since the number of iterations used for the estimation, where $\theta = 12$ and $R = 16$ ($\theta R = 192$), was too small.

Furthermore, the simulation demonstrated a bigger gain using the ADE approach for 16 actions when compared with the 8 actions case. As an example, for Row No. 1, the ADE HDPA used approximately 11,550 iterations before convergence, while the HDPA without the ADE used approximately 12,700 iterations, witch yielded a superiority of more than 1,000 iterations. Thus, the ADE had an approximately 9.95% better performance in terms of the number of iterations. In comparison, for Simulation 1, the biggest gain of using the ADE approach was approximately 4.25%.

Table 3
Experimental results for different action distributions without the ADE for Simulation 2, with 16 actions and 0.995 as the convergence criterion. The results were averaged over 100 experiments, with $\Delta = 6.75e{-}5$. The different rows represent different action configurations as described in the second column.

Row No.	Configuration characteristics	Avg	Std	Acc.
1	Config 1 in Fig.3	12,691.92	509.64	100%
2	Config 2 in Fig.3	12,362.99	413.44	100%
3	Config 3 in Fig.3	11,674.32	100.95	100%
4	Ascending	11,285.18	83.24	100%
5	Descending	11,291.90	86.09	100%

Table 4
Experimental results for different action distributions with the ADE for Simulation 2, with 16 actions and 0.995 as the convergence criterion. The results were averaged over 100 experiments, with $\Delta = 6.75e{-}5$. The different rows represent different action configurations as described in the second column.

Row No.	Configuration characteristics	Avg	Std	Acc.
1	Config 1 in Fig.3	11,556.77	111.29	100%
2	Config 2 in Fig.3	11,540.57	106.94	100%
3	Config 3 in Fig.3	11,543.75	94.47	100%
4	Ascending	11,573.89	119.07	100%
5	Descending	11,520.16	91.70	100%

Table 5
Experimental results for different action distributions without the ADE for Environments with 32 actions where 0.995 is the convergence criterion. The results were averaged over 100 experiments, with $\Delta = 4.5e{-}5$. The different rows represent different configurations as described in the second column.

Row No.	Configuration characteristics	Avg	Std	Acc.
1	As in Config 1 in Fig.3	17,690.81	849.65	100%
2	As in Config 2 in Fig.3	17,980.64	742.41	100%
3	As in Config 3 in Fig.3	16,911.9	600.69	100%
4	Ascending	15,807.09	89.47	100%
5	Descending	15,806.49	90.23	100%

5.3. Simulation 3: 32 Actions

In Tables 5 and 6, we present the results obtained for Simulation 3, which involved 32 actions. The configurations in these experiments followed the same concept as depicted in Fig. 3 for Simulation 2. We can observe that the HDPA without the ADE required considerably more iterations for the case when the optimal and sub-optimal actions were in opposite parts of the tree (Row No. 1), i.e., compared with having the actions ordered in a descending order (Row No. 5). The numbers of iterations were approximately $17,700$ and $15,800$, respectively. Comparing the results in Table 5 with those in Table 6, we observe that the ADE HDPA had a more consistent performance in terms of the number of iterations and the standard deviations. For Row No. 1 in the tables, we see that the HDPA with the ADE required approximately $16,350$, while the HDPA without the ADE required $17,700$, which is approximately 8.26% worse.

5.4. Simulation 4: 64 Actions

The configurations tested in Simulation 4 are not as *random* as those in the earlier configurations because they display a decreasing/increasing pattern in different forms. Thus, the original HDPA can benefit from the original patterns (because they already have an ordering relation rather similar to what they want to achieve with the ADE approach). As we observe in the results given in Table 7, the HDPA without the ADE had the worst performance for the configuration with randomly distributed actions in-between the optimal and sub-optimal actions being on opposite sides of the tree (Row No. 8), where it even converged to the sub-optimal action in 2% of the experiments. In Table 8, we see that the ADE HDPA required considerably fewer iterations in most cases and that the performance was consistent

Table 6
Experimental results for different action distributions with ADE HDPA for 32 actions and 0.995 as the convergence criterion. The results were averaged over 100 experiments, with $\Delta = 4.5e{-}5$ and $\theta = 12$. The different rows represent different configurations as described in the second column.

Row No.	Configuration characteristics	Avg	Std	Acc.
1	As in Config 1 in Fig.3	16,338.12	114.54	100%
2	As in Config 2 in Fig.3	16,302.25	121.44	100%
3	As in Config 3 in Fig.3	16,327.96	121.10	100%
4	Ascending	16,371.00	134.76	100%
5	Descending	16,256.67	107.27	100%

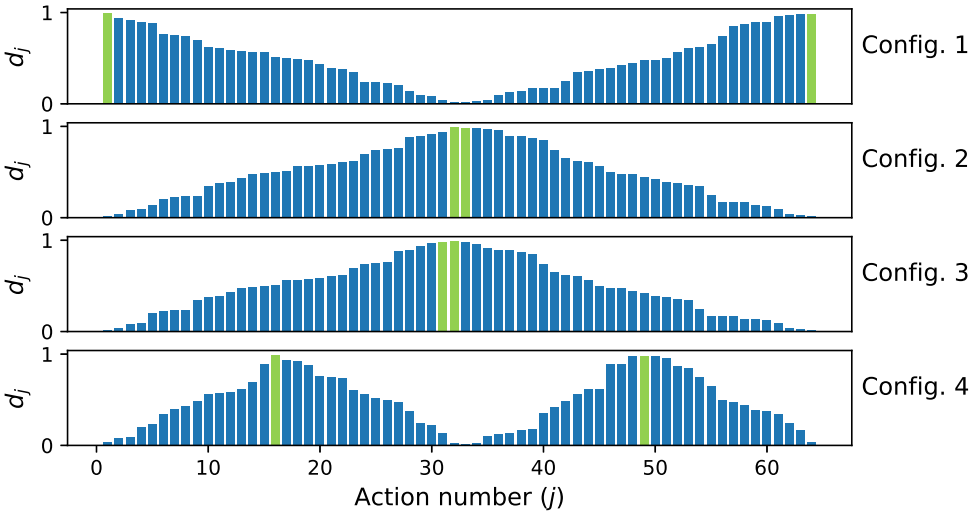

Figure 4: The figure shows some of the different configurations used in Simulation 4.

Table 7
Experimental results for different action distributions with HDPA without the ADE for 64 actions and 0.995 as the convergence criterion. The results were averaged over 100 experiments, with $\Delta = 9.9e{-}6$. The different rows represent different configurations as described in the second column.

Row No.	Configuration characteristics	Avg	Std	Acc.
1	Config 1 in Fig. 4	85,786.0	7,985.20	100%
2	Config 2 in Fig. 4	86,714.58	8,263.03	100%
3	Config 3 in Fig. 4	80,652.46	5,645.72	100%
4	Config 4 in Fig. 4	91,682.27	10,272.10	100%
5	Ascending	73,578.9	211.46	100%
6	Descending	73,527.54	199.76	100%
7	Original ordering as in [14][5]	95,220.44	11,195.3	100%
8	As in Config.1 in Fig. 3	96,201.59	11,680.31	98%

for all configurations. In this simulation, we also tried different values for θ, and as we can see, with a higher θ, the standard deviation is smaller, making the algorithm even more consistent in its performance. However, we see that a bigger θ required more iterations on average. At the same time, even for a bigger θ, the ADE HDPA is better for most cases compared with the HDPA without the ADE. For example, in Row No. 8, the HDPA with the ADE required approximately 75, 100, while the HDPA without it required approximately 96, 200, i.e, approximately 28.10% worse.

Table 8
Experimental results for different action distributions with ADE HDPA for 64 actions and 0.995 as the convergence criterion. The results were averaged over 100 experiments, with $\Delta = 9.9e{-}6$ and $\theta = 12$. The different rows represent different configurations as described in the second column.

Row No.	Configuration characteristics	Avg	Std	Acc.
1	Config 1 in Fig. 4	75,125.55	436.43	100%
2	Config 2 in Fig. 4	75,051.12	461.87	100%
3	Config 3 in Fig. 4	74,988.14	496.13	100%
4	Config 4 in Fig. 4	75,117.82	436.28	100%
5	Ascending	75,536.94	840.80	100%
6	Descending	74,716.55	408.74	100%
7	Original ordering as in [14][5]	75,073.83	426.64	100%
8	As in Config.1 in Fig. 3	75,122.10	444.24	100%
9	Ascending using $\theta = 12$	74,760.74	380.04	100%
10	Ascending using $\theta = 24$	75,255.61	237.27	100%
11	Ascending using $\theta = 100$	79,912.68	180.0	100%

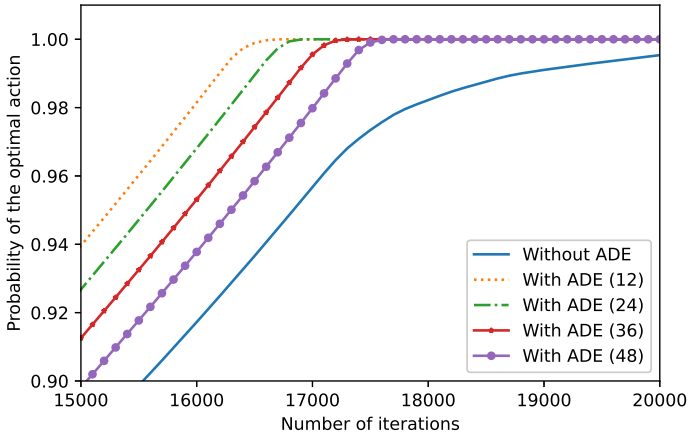

Figure 5: The figure shows the results of the experiments where the number of iterations is plotted against the probability of the optimal action. This has been done for different values of θ in Simulation 5 as configured as Config. 1 in Simulation 3. The results were averaged over 100 experiments, with $\Delta = 4.5e{-}5$.

5.5. Simulation 5: Different Settings for θ

For Simulation 5, with the results visualized in Fig. 5, we wanted to demonstrate the performance for different configurations of θ. As we can observe from the figure, the convergences of ADE with all θ-values were faster than that of the vanilla HDPA. The results in the figure are the ensemble averages over 100 experiments, where we have run each experiment for a maximum of 20,000 iterations. Indeed, we see that the number of iterations is within the range of what we would have expected from the results that we obtained in Simulation 3. Understandably, the value of θ influences the performance of the ADE, and it can be tuned according to the needs and the desired result. Clearly, the ADE approach is significantly faster than the corresponding vanilla HDPA.

5.6. Simulation 6: Different Convergence Criteria

In Fig. 6, we present the behavior of the HDPA with and without the ADE for a 32 actions problem configured as Config. 1 in Simulation 3. For different convergence criteria, we see that both the algorithm types require more iterations as the convergence criterion (T) is set higher. However, the ADE HDPA required significantly fewer iterations for all configurations. The ADE HDPA had a range in performance of approximately 16, 300 to 16, 420 in the number of iterations from 0.995 to 0.999, respectively. As opposed to this, the HDPA without the ADE had a range in performance of approximately 17, 500 to 17, 880, respectively. Consequently, the ADE HDPA required less number of iterations also

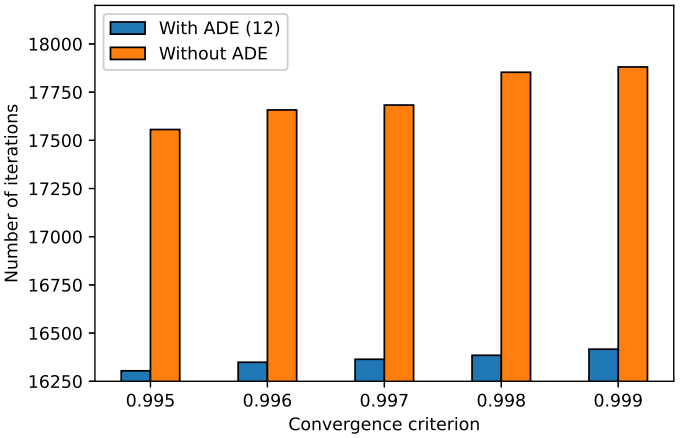

Figure 6: The figure shows results for Simulation 6 with experiments for different convergence criteria (T), and demonstrate how the convergence criteria influenced the number of iterations for HDPA with and without the ADE for 32 actions configured as Config. 1 in Simulation 3. The results were averaged over 100 experiments, with $\Delta = 4.5e{-}5$.

in the difference of iterations for the different convergence criteria. The benefit of using the ADE approach is even more superior for higher convergence criteria.

5.7. Simulation 7: Dynamic Environment

In Fig. 7, we visualize the performance of an ADE HDPA in a dynamic environment. In this simulation, we utilized Config 4 in Fig. 4, and changed the sub-optimal action ("Action 2") to be the optimal one after 20,000 iterations. Before 20,000 iterations, "Action 1" was configured as the optimal action. As we observe from the figure, the ADE HDPA dynamically adjusts its action selection vector in accordance to the optimal action changing from "Action 1" to "Action 2" throughout its operation. Thus, the machine adjusts its behavior corresponding to the change in the Environment. For the simulation visualized in the figure, the ADE HDPA converges to the optimal action ("Action 2") after around 84,000 iterations. For simulation 7, we utilized $\Delta = 9e{-}6$ and $\theta = 12$.

To compare the ADE HDPA's and plain HDPA's performance in a dynamic environment, we also averaged their number of iterations before convergence with similar parameters over 100 experiments. The methods achieved an average number of iterations of 82,882.45 and 116,665.41, respectively. The ADE HDPA achieved a 91% accuracy, and the plain HDPA achieved a 95% accuracy (converging to the sub-optimal action for the remaining percentages). These accuracies are linked with the number of iterations, and the learning parameter could be adjusted to achieve a better performance in terms of accuracy. However, the results indicate that the ADE HDPA performs better than the HDPA also in a dynamic environment, where the ADE HDPA required approximately 28% fewer iterations before attaining convergence compared with the plain HDPA[3].

5.8. Simulation 8: Comparison to the State-of-the-art Algorithms

As explained earlier, we first had traditional unstructured VSSA schemes, like the DL_{RI} scheme. After that, the discovery of pursuit and estimator algorithms followed. Thereafter, the HCPA was introduced, and later, the HDPA. In Simulation 8, we aimed to highlight the improvements these innovations represent in the field of LA, in addition to our proposed ADE approach. In Table 9, we list the results obtained from simulations conducted with eight actions as configured as Config 4 in Table 1, where 0.999 was the convergence criterion[4]. From the results in the table, we see that there is a significant improvement between the DL_{RI} scheme and the ADE HDPA approach. For the DL_{RI}, we

[3]Based on reviewer feedback, we want to emphasize that imbalanced datasets present a significant challenge in machine learning and data analysis. In such cases, the representation of certain classes within the data is disproportionately higher than others [38]. However, in the context of our research, we focus on more general scenarios, bypassing the imbalanced dataset issue. The learning automata-based channel selection algorithm proposed in this study is designed to adapt to the dynamic nature of different problems.

[4]For Simulation 8, we utilized $\Delta = 9e{-}5$ for the DL_{RI}, the HDPA, and the ADE HDPA schemes. For the HCPA and ADE HCPA, we utilized the learning parameter $\lambda = 9e{-}4$. Further, we configured $\theta = 5$ for the ADE methods.

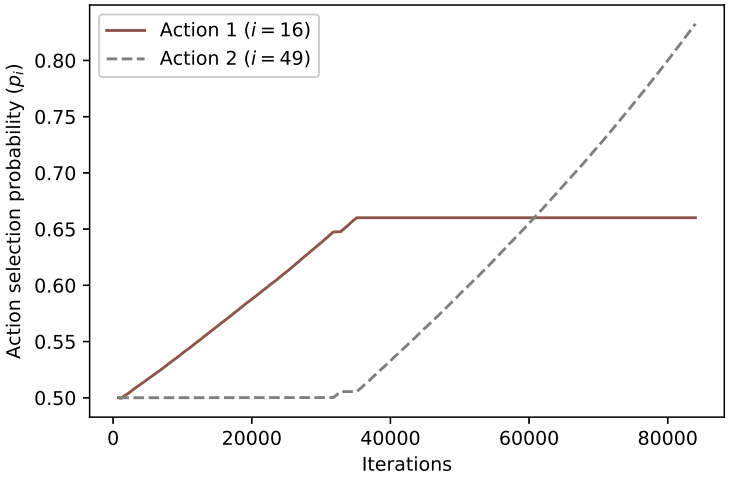

Figure 7: The development in probability for the optimal and sub-optimal actions in Simulation 7.

Parameter	DL_{RI}	HCPA	ADE HCPA	HDPA	ADE HDPA
Avg	110,933.16	7,680.24	7,490.41	7,126.38	6,834.18
Std	24,186.54	180.89	37.29	188.54	85.90
Acc.	97%	100%	100%	100%	100%

Table 9
Experimental results for different algorithms for an Environment with eight actions as configured as Config 4 in Table 1, where 0.999 was the convergence criterion. The results were averaged over 100 experiments.

even observed that the algorithm did not converge accurately. Hence, we could have utilized an even smaller learning parameter for the DL_{RI} scheme. With a smaller learning parameter, the number of iterations increases, which would have made the DL_{RI} even more inferior compared with the ADE HDPA. Furthermore, we see from the table that the ADE HDPA outperformed the other algorithms.

We also conducted simulations demonstrating the difference between HCPA, ADE HCPA, HDPA, and ADE HDPA in greater detail. The averaged results from these 100 experiments are visualized in Fig. 8. The reader should note that we could have optimized the learning parameters for the HCPA, but in this simulation, we doubled the learning rate found for the HDPA and configured $\lambda = 9e{-}4$ for the HCPA. Consequently, the learning parameter for the HDPA was configured as $\Delta = 4.5e{-}4$. The figure shows that the ADE variants reach a higher probability faster, meaning they have a more effective learning and converge faster.

5.9. Simulation 9: Communication Application

In the expanding world of wireless communication, managing network resources effectively and efficiently has become increasingly important. As the number and diversity of wireless devices continue to grow (e.g., with Internet of Things (IoT) devices), so does also the demand for better performance, throughput, and reliability in communication modules [39, 40]. Channel selection, the process of choosing the most optimal path for data transmission, plays a fundamental role in maximizing these qualities in, e.g., WiFi [41]. However, the task of channel selection has become more challenging with the introduction of heterogeneous wireless channels, where each channel's data rate may change over time due to phenomena such as channel fading [42].

Let us consider a communication module that can select one channel for communication purposes among multiple channels. The channels are heterogeneous and have different supported data rates for the communication module. Due to, e.g., channel fading, the supported data rate changes along time. Here the ADE HDPA can be applied to help the module find the best channel that has the maximum average data rate. The data rates among different channels, in positive real numbers, can be mapped into probability, namely, to numbers between 0 and 1, with a higher data rate closer to unity. In this way, the channel that has statistically higher data rate will offer higher reward probability to the

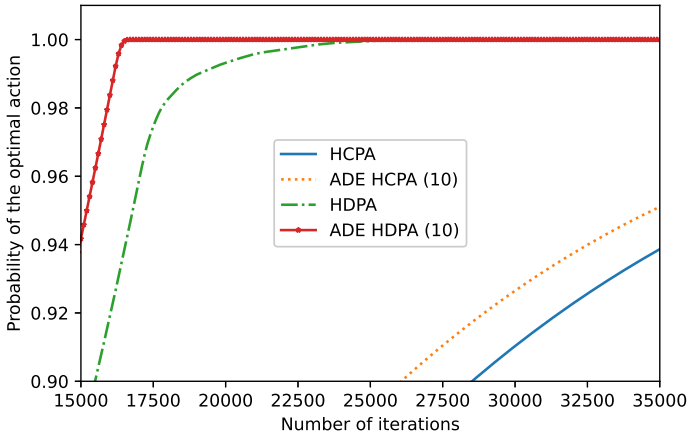

Figure 8: Experimental results for the HCPA compared with the HDPA for an Environment with 32 actions as configured as Config 1 in Fig. 3, where 0.999 was the convergence criterion. The results were averaged over 100 experiments.

LA, which will guide the LA to converge to this channel. The reader should note that this communication module, in reality, can be of any type, e.g., Bluetooth, Wifi, or LoraWAN.

To simulate the above scenario, we use 13 channels, similar to the number of channels for WiFi (e.g., 802.11b) in most countries worldwide. We simulated four APs, or four users, in the WiFi system. These users, identified by Id 1, Id 2, Id 3, and Id 4, must select the channel with the highest data rate statistically for themselves. If two or more uses happen to be in the same channel, the individual data rate is assumed to be divided by the number of users in the same channel, due to CSMA/CA. Consequently, ADE HDPA can aid them in choosing the appropriate channel. In Fig. 9, we visualize the results from this simulation. The reader should note that we can know which channel is best for particular users in a simulation environment. However, in a practical scenario, we do not have this information. In this simulation, Id 1 should select Channel 10 because this channel has a statistically higher data rate based on that user's location and channel fading characteristics. For Id 2, the optimal channel is Channel 3. For Id 3, the optimal channel is Channel 7, and for Id 4, the optimal channel is Channel 13. The figure shows that the users converged to the correct channels in around 3,500 iterations on average for the four different users.

We configured the experiments depicted in Fig. 9 to have one LA per user, and we configured the environment's stochastic responses based on a probability vector representing the statistically higher data rate for the specified user. In the simulations, we configured $\theta = 10$, and averaged the results over 100 experiments. The reader should note that, because we utilize a pursuit scheme, the LA have a more consistent learning process, and that the iterations in the start of the figure represents the additional iterations conducted in the Estimation Phase of the ADE approach. Because the results are averaged, the different users can be in between channels, but in a single experiment, it is only allowed to be inside one channel at a time. On the y-axis of the figure, we have averaged the action with the maximum action selection probability in each iteration.

5.10. Summary of the Experimental Results

The experimental results presented in this section demonstrate that the ADE HPDA has a superior performance in terms of the number of iterations and obtained standard deviation. Without the ADE approach, the performance of the HDPA can vary significantly, and the benefit of the ADE increases as the number of actions increases. Furthermore, the algorithm can perform significantly slower by randomly distributing the actions at the leaf level and not implementing the ADE approach. The ADE approach aims to order the actions at the leaf level in an ascending/descending order. However, as observed, for smaller values of θ, the ADE approach might not estimate the reward probabilities *perfectly*. Thus, the number of θ iterations used for the Estimation Phase must be set accordingly. From the rigorous simulations discussed above, we see that the ADE is a suitable solution in dynamic environments and in a communication application scenario. Additionally, the ADE HDPA requires fewer iterations before convergence compared with other earlier comparable LA algorithms, like the DL_{RI} and the HCPA.

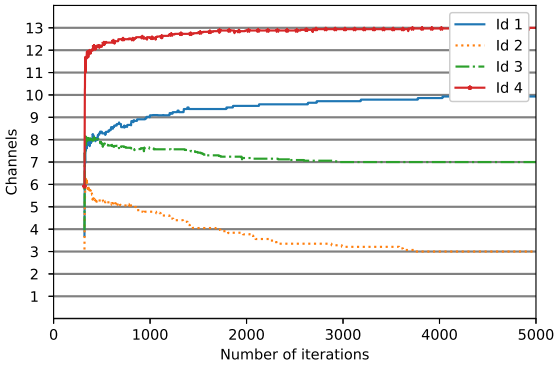

Figure 9: The figure shows results for Simulation 9 with experiments for a communication module. It demonstrates that the ADE HDPA helps the different units (identified by Id 1, Id 2, Id 3, and Id 4, respectively) select the optimal channels by learning and converging to the channel with the highest data rate for the individual units. The results were averaged over 100 experiments, with $\Delta = 4.5e{-}5$.

6. Conclusion

In this paper we have proposed the novel Action Distribution Enhancing (ADE) approach for optimally configuring the underlying hierarchical tree representing the distribution of the actions in the HCPA/HDPA. The ADE involves two phases, the first of which estimates the action probabilities very crudely, and subsequently assigns the actions at the leaves of the tree. The corresponding LA then operate in a hierarchical manner, each of them involving two actions. Our hypothesis was that if the leaves were arranged in an ascending/descending order, the collection of hierarchical automata would perform in their most optimized manner. In this paper, we have formally proven this hypothesis.

We have then proceeded to verify the power of incorporating the ADE into problems involving different numbers of automata, various Environments with corresponding reward probabilities, and for different parameter settings. Quite briefly stated, our simulation results uniformly confirm that the inclusion of the ADE significantly increases the computational accuracy and the speed of the hierarchical machine.

If we consider the chronology of LA from its infancy in FSSA through VSSA, the Estimator approaches, and the more-recent hierarchical schemes, we modestly believe that the inclusion of the ADE represents the state-of-the-art which will not be able to be surpassed too easily.

References

[1] K. D. Abeyrathna, B. Bhattarai, M. Goodwin, S. Gorji, O.-C. Granmo, L. Jiao, R. Saha, and R. K. Yadav, "Massively Parallel and Asynchronous Tsetlin Machine Architecture Supporting Almost Constant-Time Scaling," in *The Thirty-eighth International Conference on Machine Learning (ICML 2021)*, ICML, 2021.

[2] 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 of Things Journal*, vol. 6, no. 4, pp. 6007–6017, 2019.

[3] X. Deng, Y. Jiang, L. T. Yang, L. Yi, J. Chen, Y. Liu, and X. Li, "Learning-automata-based confident information coverage barriers for smart ocean internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9919–9929, 2020.

[4] O.-C. Granmo, "The Tsetlin Machine – A Game Theoretic Bandit Driven Approach to Optimal Pattern Recognition with Propositional Logic," April 2018, arXiv:1804.01508. [Online]. Available: http://arxiv.org/abs/1804.01508

[5] H. Guo, S. Li, K. Qi, Y. Guo, and Z. Xu, "Learning automata based competition scheme to train deep neural networks," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 2, pp. 151–158, 2020.

[6] 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.*, accepted, Aug. 2022.

[7] 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 Communications Letters*, vol. 8, no. 1, pp. 69–72, 2019.

[8] S. Lakshmivarahan and M. A. L. Thathachar, "Absolutely Expedient Learning Algorithms for Stochastic Automata," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, no. 3, pp. 281–286, 1973.

[9] S. Lakshmivarahan, *Learning Algorithms Theory and Applications*, ed. 1, New York: Springer-Verlag, 1981.

[10] J. K. Lanctot and B. J. Oommen, "Discretized Estimator Learning Automata," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 6, pp. 1473–1483, 1992.

[11] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*, Dover Books on Electrical Engineering Series, Dover Publications, Courier Corporation, 2013.

[12] D. T. H. Ng, B. J. Oommen and E. R. Hansen, "Adaptive Learning Mechanisms for Ordering Actions Using Random Races," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 5, pp. 1450-1465, 1993.

[13] Narendra, Kumpati S., and Mandayam A. L. Thathachar, "Learning Automata: An Introduction.", Illustrated edition, Dover Publications, 2012.

[14] R. O. Omslandseter, L. Jiao, X. Zhang, A. Yazidi and B. J. Oommen, "The Hierarchical Discrete Pursuit Learning Automaton: A Novel Scheme With Fast Convergence and Epsilon-Optimality," *IEEE Transactions on Neural Networks and Learning Systems*, December 2022.

[15] Goodfellow, Ian, et al., "Deep Learning", MIT Press, 2016.

[16] H. Shen, "Towards a Mathematical Understanding of the Difficulty in Learning with Feedforward Neural Networks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 811-820, 2018.

[17] C. Sun, A. Shrivastava, S. Singh and A. Gupta, "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 843-852, Italy, 2017.

[18] R. O. Omslandseter, L. Jiao, Y. Liu, and B. John Oommen, "User grouping and power allocation in NOMA systems: A novel semi-supervised reinforcement learning-based solution," *Pattern Analysis and Applications*, July 2022.

[19] B. J. Oommen, "Absorbing and Ergodic Discretized Two-Action Learning Automata," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 2, pp. 282–293, 1986.

[20] B. J. Oommen and M. Agache, "Continuous and Discretized Pursuit Learning Schemes: Various Algorithms and Their Comparison," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 31, no. 3, pp. 277–287, 2001.

[21] 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 Transactions on Computational Social Systems*, vol. 7, no. 4, pp. 1004–1018, 2020.

[22] S. Sahoo, B. Sahoo, and A. K. Turuk, "A Learning Automata-Based Scheduling for Deadline Sensitive Task in The Cloud," *IEEE Transactions on Services Computing*, vol. 14, no. 6, pp. 1662–1674, Nov. 2021.

[23] 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 Systems Journal*, vol. 13, no. 1, pp. 313–323, 2019.

[24] 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.

[25] M. A. L. Thathachar and P. S. Sastry, "Estimator Algorithms for Learning Automata," *Proceedings of the Platinum Jubilee Conference on Systems and Signal Processing*, Department of Electrical Engineering, Indian Institute of Science, 1986.

[26] M. L. Tsetlin, "Finite Automata and Modeling the Simplest Forms of Behavior," *Uspekhi Matem Nauk*, vol. 8, no.4, pp. 1–26, 1963.

[27] M. L. Tsetlin, *Automaton Theory and the Modeling of Biological Systems*, New York: Academic Press, 1973.

[28] R. K. Yadav, L. Jiao, O.-C. Granmo, and M. Goodwin, "Robust interpretable text classification against spurious correlations using and-rules with negation," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, July 2022, pp. 4439–4446.

[29] Z. Yang, Y. Liu, Y. Chen, and L. Jiao, "Learning automata based q-learning for content placement in cooperative caching," *IEEE Transactions on Communications*, vol. 68, no. 6, pp. 3667–3680, 2020.

[30] 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 Transactions on Neural Networks and Learning Systems*, vol. 31, no. 2, pp. 512–526, 2020.

[31] 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 Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3444–3457, 2021.

[32] X. Zhang, O.-C. Granmo, B. J. Oommen, "Discretized Bayesian Pursuit - A New Scheme for Reinforcement Learning," *Advanced Research in Applied Artificial Intelligence*, vol. 7345, pp. 784–793, 2012.

[33] X. Zhang, O.-C. Granmo and B. J. Oommen, "On Incorporating the Paradigms of Discretization and Bayesian Estimation to Create a New Family of Pursuit Learning Automata," *Applied Intelligence*, vol. 39, no. 4, pp. 782-792, 2013.

[34] 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 Analysis and Applications*, vol. 20, pp. 797–808, 2017.

[35] 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, 2022.

[36] Z. Zhang, D. Wang, and J. Gao, "Learning automata-based multiagent reinforcement learning for optimization of cooperative tasks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 10, pp. 4639–4652, 2021.

[37] 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. 81 236–81 252, 2021.

[38] Alam TM, Shaukat K, Khan WA, Hameed IA, Almuqren LA, Raza MA, Aslam M, and Luo S. "An Efficient Deep Learning-Based Skin Cancer Classifier for an Imbalanced Dataset," *Diagnostics (Basel)*, 2022.

[39] J. G. Andrews et al., "What Will 5G Be?," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, 2014.

[40] P. Kamble and A. N. Shaikh, "6G Wireless Networks: Vision, Requirements, Applications and Challenges," *2022 5th International Conference on Advances in Science and Technology (ICAST)*, pp. 577–581, 2022.

[41] D. Saliba, R. Imad, and S. Houcke, "Wifi channel selection based on load criteria," *2017 20th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pp. 332–336, 2017.

[42] X. Zhang, L. Jiao, O. -C. Granmo, and B. J. Oommen, "Channel selection in Cognitive Radio Networks: A Switchable Bayesian Learning Automata approach," *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 2362-2367, 2013.