The Hierarchical Continuous Pursuit Learning Automation: A Novel Scheme for Environments with *Large* Numbers of Actions

Anis Yazidi, Xuan Zhang, Lei Jiao, and B. John Oommen, *Life Fellow, IEEE*

***Abstract*—Although the field of Learning Automata (LA)[1] has made significant progress in the last four decades, the LA-based methods to tackle problems involving environments with a large number of actions is, in reality, relatively unresolved. The extension of the traditional LA to problems within this domain cannot be easily established when the number of actions is very large. This is because the dimensionality of the action probability vector is correspondingly large, and so, most components of the vector will soon have values that are *smaller* than the machine accuracy permits, *implying that they will never be chosen*. This paper presents a solution that extends the continuous pursuit paradigm to such *large*-actioned problem domains. The beauty of the solution is that it is hierarchical, where all the actions offered by the environment reside as leaves of the hierarchy. Further, at every level, we merely require a *two*-action LA which automatically resolves the problem of dealing with arbitrarily small action probabilities. Additionally, since all the LA invoke the pursuit paradigm, the best action at every level trickles up towards the root. Thus, by invoking the property of the "max" operator, in which, the maximum of numerous maxima is the overall maximum, the hierarchy of LA converges to the optimal action. This paper describes the scheme, and formally proves its ε-optimal convergence. The results presented here can, rather trivially, be extended for the families of discretized and Bayesian pursuit LA too. The paper also reports extensive experimental results (including for environments having 128 and 256 actions) which demonstrate the power of the scheme and its computational advantages. *As far as we know, there are no comparable Pursuit-based results in the field of LA*. In some cases, the HCPA requires less than 18% of the number of iterations than the benchmark L_{R-I} scheme, *which is, by all metrics, phenomenal*.**

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

Author's status: *Professor*. This author can be contacted at: Oslo Metropolitan University, Department of Computer Science, Pilestredet 35, Oslo, Norway. E-mail: anisy@oslomet.no.

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

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

Author's status: *Chancellor's Professor*; *Life Fellow: IEEE* and *Fellow: IAPR*. This author can be contacted at: School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6. This author is also an *Adjunct Professor* with the University of Agder in Grimstad, Norway. E-mail address: oommen@scs.carleton.ca.

[1]The work of the last author was partially supported by NSERC, the Natural Sciences and Engineering Council of Canada. A preliminary version of this paper was published in the *Proceedings of AIAI'2018, the 2018 International Conference on Artificial Intelligence Applications and Innovations*, held in Rhodes, Greece, in May 2018. We are very grateful for the feedback from the anonymous Referees of the *original* submission. Their input significantly improved the quality of this final version.

I. Introduction

In this paper, we deal with field of the Learning Automata (LA) [2]. The area of LA has been initiated for several decades ago by Tsetlin [54] who pioneered the first learning models able to operate in random environments. The work of Tsetlin and subsequent development in the field LA has served as the precursor for the area of reinforcement learning. In contrast to other fields within the area of Artificial Intelligence (AI) in which the Environment is deterministic and consistently provides the same answer to the same query, an LA interacts in general with a random Environment where the "Teacher" can provide different random responses to the same query over time. An LA is by definition a decision making mechanism that has a learning capability. The aim of the LA is to sequentially learn the optimal action among a set of actions provided by the Environment. In general, the Environment is usually stochastic. At each time instant, an action is chosen by the LA and given as input to the Environment. The Environment in turn returns an output which is the response to the action provided by the LA, this response is usually a Reward or Penalty. Based on both the response of the Environment and the internal state of the LA, the LA adjusts its action selection strategy in a deterministic or stochastic manner. This adjustment is hopefully designed carefully so that the LA converges over to the optimal action with both high accuracy and speed.

Brief overview of LA: Initial LA were designed to be Fixed Structure Stochastic Automata (FSSA), whose state update and decision functions are time invariant [32, 54]. Later, Variable Structure Stochastic Automata (VSSA), such as 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 ($L_{R-\varepsilon P}$) scheme [25, 32] (which are characterized by *functions* that update the action selection probabilities), were developed. Schemes which invoke nonlinear functions have also been designed and analyzed [25, 26, 32]. These updating functions can be either continuous or discretized, where the latter are, generally speaking, faster than their continuous counterparts [27, 28, 35, 36, 39, 39, 50, 63]. Further, the Markovian representation of the LA can be

[2]The initial version of this paper had a more detailed overview of the field including FSSA, VSSA, the Continuous/Discretized families, and the Ergodic/Absorbing families. We are grateful to the anonymous Referee who requested abridging it with the fair assumption that the reader will be familiar with the fundaments of LA. However, comprehensive surveys are available in [25, 31, 32, 33, 44, 49].

either absorbing or ergodic [25, 31, 32, 33, 35, 44], where the latter are known to better adapt to non-stationary environments where the reward probabilities are time dependent.

Estimator-based LA: The fastest LA to-date are those which also involve the estimates of the reward probabilities, and these are referred to as Estimator Algorithms (EAs). They work with a noticeably different paradigm, namely one in which the phases of "Exploration" and "Exploitation" are continuously and constantly interleaved. During each learning cycle, these algorithms incorporate an estimation phase, in which they update the estimates of the reward probabilities of the various actions, thus maintaining the so-called "Estimator" vector. These LA, pioneered by Thathachar and Sastry [52], render the learning process to be more goal-directed, and the probability updating of the action probability vector involves an updating function and the "Estimator" vector. This, in turn, also leads to a much faster convergence – almost an order of magnitude faster than the continuous and discretized VSSA. More recently, PAs that use the Bayesian estimates (instead of the ML estimates) in the "Estimator" vector have also been designed and analyzed. They probably constitute the fastest LA to-date [64, 67]. This present paper further enhances the field of EAs, and in particular, the *Pursuit* Algorithms (PAs) described below.

Pursuit-based LA: Within the family of EAs, the set of *Pursuit* Algorithms (PAs) were the pioneering schemes, whose design and analysis, as mentioned earlier, were initiated by Thathachar and Sastry [52]. Until the last few years, the "Estimator" vector contained the Maximum Likelihood (ML) estimates of the actions' reward probabilities. The first Pursuit Algorithm (PA) was designed to operate by updating the action probabilities based on the L_{R-I} paradigm. In each iteration, the PA determines the current "Best" action based on the estimates of the reward probabilities, and then pursues *the "Best" action* by linearly increasing *its* action probability. The Continuous Pursuit Algorithm (CPA), which invokes the L_{R-I} updating paradigm, was the pioneering member of these EAs. Oommen and Lanctot [39] presented the Discretized Pursuit Algorithm (DPA) by discretizing the action probability space. The DPA was shown to be superior to its continuous counterpart. PAs have also been extended by allowing them to be of the Reward-Penalty paradigms [36]. All the families of the reported Pursuit LA (please see [1, 62, 63, 64]) have been shown to be faster than VSSA. Besides, faster and more efficient discretized versions of all the reported EA schemes have also been devised [36, 64].

Proofs of Convergence for LA: The most difficult part in the design and analysis of LA consists of the formal proofs of their convergence accuracies. Of these, understandably, the most difficult proofs involve the family of EAs. This is because the convergence involves two intertwined phenomena, i.e., the convergence of the reward estimates *and* the convergence of the action probabilities themselves. Ironically, the *combination* of these in the updating rule is what renders the EA fast. However, if the accuracy of the estimates are poor because of inadequate estimation (i.e., the sub-optimal actions are not sampled "enough number of times"), the convergence accuracy can be diminished, which is really a dilemma! The original proofs of convergence of PAs, which erroneously invoked the monotonicity property, have been rectified to use the martingale property in [1, 62, 63, 64].

Applications of LA: With regard to applications, the entire field of LA and stochastic learning has had a myriad of applications [25, 31, 32, 44, 49], which (apart from the many applications listed in these books) include solutions for problems in network and communications [30, 34, 43], network call admission, traffic control, quality of service routing [2, 3, 57], distributed scheduling [47], training hidden Markov models [24], neural network adaptation [29], intelligent vehicle control [55, 56], service selection [60] and even fairly theoretical problems such as graph partitioning [37] and string taxonomy [38]. Besides these fairly generic applications, with a little insight, LA can be used to assist in solving (by, indeed, learning the associated parameters) the stochastic resonance problem [12], the stochastic sampling problem in computer graphics [13], the problem of determining roads in aerial images by using geometric-stochastic models [7], and various location problems [9]. Similar learning solutions can also be used to analyze the stochastic properties of the random waypoint mobility model in wireless communication networks [8], to achieve spatial point pattern analysis codes for GISs [46], to digitally simulate wind field velocities [40], to interrogate the experimental measurements of global dynamics in magneto-mechanical oscillators [16], and to analyze spatial point patterns [6] – to mention a few other applications.

Other related Research Directions: There are various alternate direction of research that are closely to the field of LA, and one of them involves the Multi-Armed Bandit Problem (MABP)[3]. The MABP problem consists of working with multiple "arms", and "pulling" each of them yields a "reward" with some unknown probability. The goal is to determine the optimal arm that is to be pulled at each trial, so as to maximize the rewards within a time span.

Classical exact solutions to the MABP for discounted rewards rely on invoking on the so-called Gittins Indices [19, 20, 59], i.e., by always pulling the arm associated with the largest Gittins index. However, calculating these Gittins Indices in a computationally efficient manner is not an easy task. Consequently, a number of approximate solutions to the MABP have been proposed. The *e*-greedy strategy [58], is one of the early examples. In this strategy, the parameter ε is essential to balance between the exploration and the exploitation phases of the learning exercise. Here, the arm that is thus far *being perceived* as being the best is pulled with probability $1-\varepsilon$, and a randomly chosen action is pulled with probability ε. However, the latter scheme is not absorbing as in the case of LA algorithms that are mainly designed for stationary environments. Consequently, sub-optimal actions will be chosen with strictly positive probability, unlike in the field of LA, where the probability mass associated to non-optimal actions asymptotically tends to zero. Variants of *e*-greedy strategy include the *e*-decreasing strategy [4, 11], and

[3]We are very grateful to the anonymous Referee who requested this section. The relationship between the field of LA and MABP has been well recorded in the literature. One should note, however, that the methods used in solving them and the metrics for the respective fields are quite distinct.

the *e*-greedy strategy based on reward Value Differences (VDs) [53]. All these gradually decrease the losses as they try to shift the scheme's focus from exploration to exploitation even as the trial process proceeds. The ultimate goal, of course, is that of obtaining a better balance between the exploration and exploitation phases. Furthermore, from a philosophical perspective, the mathematical tools used in the field of LA are totally different from those used in bandit problems. In bandit problems, the main focus is to usually bound the maximal regret which is the loss induced by choosing the non-optimal actions, deduced over a finite time horizon. In the field of LA, the main focus is the asymptotic convergence of the scheme where the main tool is the theory of martingales.

Families of approximate solutions include confidence-interval based algorithms useful in similar learning settings. These algorithms estimate confidence intervals for the reward probabilities, and identify an "optimistic" estimate of the reward probability for each arm. The arm with the most optimistic reward probability estimate is then greedily selected. Auer *et al.* [4] have shown that variants of confidence interval based algorithms, the so-called UCB and UCB-Tuned schemes, provide a logarithmic regret bound.

Few notable solutions for handling large numbers of actions in bandit problems are found in the literature. A well-known scheme [17] uses the idea of "median" elimination, where the estimates of the arms' rewards are updated in epochs composed of a number of iterations, and in each round, arms with reward estimates that are less than the median arm are eliminated. The algorithm falls under the class of *Probably Approximately Correct* (PAC) learning. It is clear that this algorithm has a different objective from our LA schemes. Being PAC, it can eliminate the optimal arm at an early stage.

When it comes to multi-armed bandit problems involving a large number of actions and which use tree-based solutions, we identify two main families of solutions. The first family which fueled a lot of research interest is called the Monte-Carlo Tree Search (MCTS) bandit solution due to Kocsis *et al.* and which was extended in several studies such as [14, 15]. The algorithm uses a tree-structured search space and extends the UCB algorithm proposed by Auer *et al.* [4]. The intuitive idea is to dynamically grow a tree and to bias its growth towards the most promising regions which reduces the search space. However, the algorithm is more suitable to games where a path in a tree represents a sequence of moves of different players, and it has thus found applications in computerized games as for "Go" [18]. The algorithm proposed in this paper cannot be compared to this class of MCTS-based algorithm because in our solution, the reward of a path in the tree, corresponds to the reward obtained at the leaf node and not to the sum of rewards along the path.

A second family of work involving trees is reported in [10] and involves the concept of Hierarchical Optimistic Optimization [10]. The latter work also deals with a large set of actions but in a continuous space. Here rewards are maintained for each subtree. Since the numbers of actions are infinite and correspond to the interval, the tree grows in regions with high rewards in order to construct estimates at a higher resolutions over the line in the promising regions. The work can be considered as a class of stochastic global optimizations with a Lipschitz reward function that has to be optimized.

The primary difference between the strategies described above and the LA-based methods are that the latter (within the VSSA category) use the action probability vector to choose the actions. Thus, the more optimal ones, with higher reward probabilities will, as time passes, be chosen more often. In this way, inferior actions are chosen with decreasing likelihood, implying that the estimates and other characterizing factors (such as the above-mentioned indices) need not be computed so frequently for these. The metrics that quantify the performance of the algorithms in the respective fields are also different. Thus, to ensure that this current paper is streamlined, we have chosen to restrict ourselves to methodology, metrics and comparisons of the LA-based methods.

A. Problem Statement: LA when R is large

Devising LA for specially dealing with a large number R of actions is a very pertinent LA problem but in the same time it is one of the hardest problems, for the reasons that we enumerate here:

1) In the case of FSSA, one requires N-states for each of the R actions. As the environment responds, the LA, for the most part (i.e., except at the so-called boundary states) moves within the states of a single action, and it can take a large number of iterations (for example, in the Krinsky LA) for the machine to even *enter* the boundary state of another action. Before all the actions are even visited, in certain environments, it could take tens of thousands of iterations for all the actions to be visited.
2) In the case of FSSA, since the machines are almost always ergodic, the Markov chain lingers in its transient behavior for a long period of time before convergence. In this case, when the number of actions is large, one deals with an $R \cdot N \times R \cdot N$-sized Markov chain, and this adds to the sluggishness of the machine.
3) In the case of VSSA, the above two concerns are mitigated by the use of the action probability vector. This has noticeable advantages, when R is relatively small, for example, of the order of 10. In this case, the action probability vector has the dimension R, and all the actions have a reasonable probability of being chosen. This permits the LA to discriminate between the various actions, and to converge to the superior one. However, when R becomes high, most of the action choice probabilities can have very small values and may not even be chosen, thus rendering the principle behind VSSA to be void.
4) In the context of VSSA, for example, in the linear scheme, the probabilities which are decreased are multiplied by a constant. Thus for any R, typically $R-1$ of these probabilities may have to be decreased. Notice that when R is large, the decrement of these $R-1$ probabilities can make a "non-small-step" change in the probability that is being increased. This will, consequently, significantly hinder the convergence of the

machine, inasmuch as all the convergence proofs depend on the theory of "small-step" random process. The same assertion is valid for the discretized families of LA.

5) The families of continuous and discrete pursuit algorithms, described above, are universally accepted to be the fasted reported LA. This is because, as mentioned above, they augment the action probability vector with a vector of the estimates of the reward probabilities. When R is large, this poses a problem of a disproportionate magnitude, because all the R actions have to be sampled a reasonably large number of times so that the inferior actions can be filtered out. Thus, pursuit LA are also quite sluggish when, R, the number of actions are large.
6) Hierarchical systems of LA have also been studied in [51], where the authors presented a complexity analysis and showed that the maximum computational saving is obtained if the number of actions for each LA in the hierarchical system is either 2 or 3. Analogous hierarchies were also discussed in [32] and [41], where the basis of the schemes was the use of "traditional" LA at every level (parent and sibling) of the hierarchy. That being said, the uniqueness of the present scheme is that unlike the previous works, in this paper, we have used "Pursuit"-based LA in the hierarchy. The consequence of this is that we do not have to wait till the various action probability vectors at the various levels converge at every level, to "trickle" the solution up the hierarchy. Rather, we can utilize the property of the "Max" operator to choose the currently superior estimated action at every level, and merely pursue *its* action probability. All of these issues will be clear in the subsequent sections.

The solution that we advocate in this article adresses all the aforementioned issues.

B. Contributions of the Paper

We summarize the contributions of this papers as follows:

1) We suggest a novel hierarchical LA solution which uses a tree structure as a part of the learning process. Unlike the main stream of LA solutions, we do not use the FSSA or VSSA to design the learning.
2) Our scheme is based on multi-level hierarchy composed of two actions CPA LA at each of the levels. Interestingly, both the estimation and interaction *take solely place at the leaf of the hierarchy*.
3) We propose a manner by which individual LA perform the learning locally and then the estimates are trickled-up in recursive manner by *only* considering a node and its sibling in order to achieve global learning.
4) Our scheme solves the problem of having actions probabilities below machine accuracy. Our estimates are manageable to accomplish even with a low machine accuracy.
5) The convergence speed of this novel LA proposed in this paper is *many* order magnitude faster than any other legacy LA. Therefore, we establish that the proposed LA improves over convergence accuracy and speed which are two properties that are known to be hard to jointly improve since improving one affects usually negatively the other one. We have tested our schemes for an environment compromising large numbers of actions: 128 and 256 actions. To the best of our knowledge, those experimental results document the largest number of actions deployed in any LA study reported in the literature.

C. Organization of the Paper

First of all, in Section II we describe the above-mentioned HCPA in detail. Section III then proves the convergence of the scheme. The experimental results and the comparison of its performance in benchmark Environments is presented in Section IV. Section V concludes the paper.

II. The HCPA LA

A. Rationale for Our Solution

The rationale for our solution is akin to the philosophy behind the acclaimed binary search paradigm. If we have to search for a record in a list of unsorted records, it will require a linear number of searches. However, if there is a mechanism by which half the records can be discarded, for example, in a sorted list, the number of probes reduces to be logarithmic. This is, precisely, what we shall endeavor to do.

The philosophy motivating our new scheme resorts to superimposing the actions onto a binary tree[4], in which, the leaves are the actual actions themselves. Further, each internal node represents the best action in the *entire subtree* below that node. By performing comparisons between the actions in a pairwise manner, i.e., at the leaves of the tree, only the superior actions are trickled up towards the root. By doing this, one always deals with 2-action LA. Here, however, unlike the work of previous researchers [5], we do not resort to FSSA or traditional VSSA, to differentiate between the various pairs of actions at the leaves. Rather, we shall use the 2-action continuous pursuit LA [66]. Since $R = 2$ at every level, the number of iterations required to achieve the estimation is considerably less. Further, the estimation that is achieved at the leaf level, is all that is required for the entire tree – no estimation operations are required at the internal nodes.

A notable attempt to devise hierarchical LA is due to Papadimitriou [42]. Before we comment on this work, we mention that the Pursuit concept can be used in a Continuous or Discretized paradigm, and that the action probabilities can be changed on Reward-Penalty (RP), Reward-Inaction (RI) and Inaction-Penalty (IP) scenarios. Consequently, we would have six Pursuit variants: CP_{RP}, DP_{RP}, CP_{RI}, DP_{RI}, CP_{IP} and DP_{IP}, and among these, Agache and Oommen [36] has reported that the DP_{RI} is the most performant scheme. Papadimitriou in his seminal work on hierarchical LA [42] has indeed used this latter machine, and this is meritorious. Nevertheless, the differences between our work and the work due to Papadimitriou [42] are fundamental and notable. The first major difference revolves around the tree model and the

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

placement of the actions. The second difference lies in the way we trickle up and propagate the estimates at different levels of the tree by considering the concept of the "maximum" among the siblings, and by this manner, we do not need to probe the environment at each iteration for efficiently updating the estimates. Furthermore, the interactions only take place at the leave nodes. Furthermore, our proof is completely different - it asserts the submartinagle property at every node. Because of this, our scheme is superior to the state-of-the-art LA schemes and this was further demonstrated through a large experiment including more actions than any other reported stduies – of even 256 actions [5]

All these issues will be clarified presently.

B. Construction of the Hierarchy

We shall explain the way according to which the search space is constructed. The hierarchy is organized as a balanced full[6] binary tree with maximal depth K. In order to simplify the notation and to ease the formal description of the scheme, we will adhere the notation used in [23, 61], and use jointly two indexes to refer to a node in a tree, one index related to the depth and one index related to the relative order of the node in question among the nodes located at the same tree depth. We shall give the details of the hierarchy as follows.

1) **Root node**: At depth 0 lies one single LA that corresponds to the root of the hierarchy.
2) **The various LA**: To each node in the tree, we attach a a 2-action LA $\mathcal{A}$, with actions denoted as 0 and 1.
3) **Activations of LA for K levels: from 0 to K-1**:
   - **The different LA at depth** k: The LA $j \in \{1,...,2^k\}$ at depth k, is denoted by $\mathcal{A}_{\{k,j\}}$, where $0 \leq k < K-1$, and it possesses two actions $\alpha_{\{k+1,2j-1\}}$ and $\alpha_{\{k+1,2j\}}$.
     - In the case where the action $\alpha_{\{k+1,2j-1\}}$ is selected, the LA $\mathcal{A}_{\{k+1,2j-1\}}$ becomes active.
     - In the case where the action $\alpha_{\{k+1,2j\}}$ is selected, the LA $\mathcal{A}_{\{k+1,2j\}}$ becomes active.
     - Informally stating, $\mathcal{A}_{\{k+1,2j-1\}}$ and $\mathcal{A}_{\{k+1,2j\}}$ represent the *Left Child* and *Right Child* of their parent LA $\mathcal{A}_{\{k,j\}}$ respectively.
   - **The LA at depth** $K-1$: The LA residing at depth $K-1$, i.e., one level *just* above the leave nodes, is responsible for choosing the action from the stochastic environment.
     - This LA in question has the two actions $\alpha_{\{K,2j-1\}}$ and $\alpha_{\{K,2j\}}$.
     - At this level $K-1$, the total number of actions is 2^K actions: $\alpha_{\{K,j\}}$ where $j \in \{1,...,2^K\}$ at depth K.
     - Please note that $\alpha_{\{K,j\}}$ is associated with its "parent LA" $\mathcal{A}_{\{K-1,\lceil j/2 \rceil\}}$.
4) **At level K**: Finally, the nodes residing at depth K, which is the maximal depth of the tree, do not have children.

C. The Proposed Solution

Based on the above description of the hierarchy and the tree, the HCPA works as described below.

At the bottom-most level, we resort to a two-action CPA to identify the superior action among the two siblings actions at this level. To achieve this goal, we resort to two pieces of information, a two-dimensional estimate vector and the two-action probability vector from the last time step in order to update the current probability vector. The maximum of these estimates is trickled to the parent, and this newly adopted estimate by the parent is compared to the estimate of the sibling of this parent. The procedure is continued recursively until reaching the root node where each time the estimate is update and the probability vector is updated too.

D. The Proposed Solution: An Example

By way of example, consider Figure 1 which involves a three-level problem.

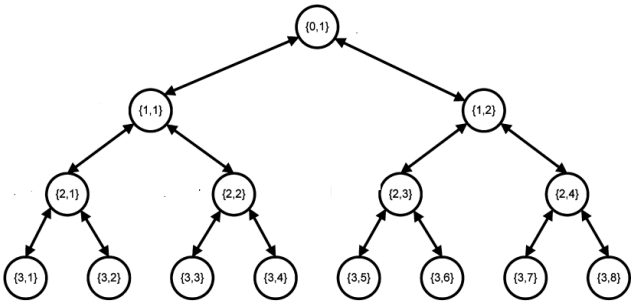

Fig. 1: The operation of the HCPA, where each node is an LA. The leaf nodes are the real actions that directly interact with the Environment itself.

These 8 actions are at the leaves of the tree, and are denoted by $\{3,1\},\ldots,\{3,8\}$. At the second level, we have the LA denoted by $\{2,1\},\ldots\{2,4\}$. Consider the operation of the LA denoted by $\{2,1\}$. For its operation, the estimates of the reward probabilities of the actions $\{3,1\}$ and $\{3,2\}$ are first obtained, and the CPA performs the corresponding updating rule. Let us assume that the reward probability estimate of $\{3,1\}$ is the superior estimate. This value is then trickled up to $\{2,1\}$. Similarly, if the reward probability estimate of the action $\{3,4\}$ is greater than that of $\{3,3\}$, the estimate of $\{3,4\}$ is trickled up to $\{2,2\}$. The reward probability estimates of the actions $\{2,1\}$ and $\{2,2\}$ are compared, and the larger one is trickled up to $\{1,1\}$. Informally, one can easily see that if this process is done recursively and correctly, the LA represented by the root will converge if the CPA uses a parameter that is arbitrarily close to zero. In this case, as we shall prove

[5] In [42], Papadimitriou has conducted experiments with a maximum of 64 actions. Unfortuantely, we were not able to perform a fair comparison between our scheme and the work done in [42] because the authors does not state the size of the ensemble runs he tested [42]. In this paper, we use an ensemble as large as 400, and we opted for the best parameter that achieved "absolute" convergence which means correct convergence in each single run. However, we should applaud the work [42] as it is the first work that pioneered the idea of hierarchical Pursuit-based LA !!

[6] It is easy to generalize the structure by assuming that whenever the number of actions is less than 2^K, some dummy actions can be added to the hieacry to round up the number and those dummy actions are constucted using reward probability zero.

formally, the LA at the root will converge to the overall best action with an arbitrarily high probability.

The process described informally here is formalized in the Algorithm below, and its convergence properties are proven in Section III. The experimental proofs of its superiority to the various families of LA is demonstrated in Section IV.

E. The Algorithm of the Proposed Solution

1) Notation and Definitions: At this juncture, we shall present our notation:

- The 2^K actions that interact with the Environment lie in the set $\left\{\alpha_{\{K,1\}},\ldots\alpha_{\{K,2^K\}}\right\}$. Further, the actions $\left\{\alpha_{\{K,2j-1\}},\alpha_{\{K,2j\}}\right\}$ are the two only possible actions to choose among at level $K-1$, namely $\mathcal{A}_{\{K-1,j\}}$.
- Each LA $j \in \{1,...,2^k\}$ at depth k, called $\mathcal{A}_{\{k,j\}}$, where $0 \leq k \leq K-1$ has two actions, namely, $\alpha_{\{k+1,2j-1\}}$ and $\alpha_{\{k+1,2j\}}$.
- $P_{\{k,j\}} = [p_{\{k+1,2j-1\}}, p_{\{k+1,2j\}}]^T$ corresponds to the action probability vector of LA $\mathcal{A}_{\{k,j\}}$, where $0 \leq k \leq K-1$.

Begin Algorithm HCPA

Parameters:

λ: The learning parameter, where $0 < \lambda < 1$, where λ is close to zero.

$u_{\{K,2j-1\}}$, $u_{\{K,2j\}}$: The number of times $\alpha_{\{K,2j-1\}}$, $\alpha_{\{K,2j\}}$ have been rewarded *when* it has been selected.

$v_{\{K,2j-1\}}$, $v_{\{K,2j\}}$: The number of times $\alpha_{\{K,2j-1\}}$, $\alpha_{\{K,2j\}}$, has actually been selected.

$\hat{d}_{\{K,2j-1\}}$, $\hat{d}_{\{K,2j\}}$: The estimate of the reward probabilities of $d_{\{K,2j-1\}}$, $d_{\{K,2j\}}$, computed as:

$\hat{d}_{\{K,2j-1\}} = \frac{u_{\{K,2j-1\}}}{v_{\{K,2j-1\}}}$, $\hat{d}_{\{K,2j\}} = \frac{u_{\{K,2j\}}}{v_{\{K,2j\}}}$.

$\hat{D}$ is the vector of the estimates $\{\hat{d}\}$.

m: The index of the optimal action.

h: The index of the greatest element of $\hat{D}$.

R: The response from the Environment, where $R=0$ corresponds to a Reward, and $R=1$ to a Penalty.

T: A Threshold, where $T \geq 1-\varepsilon$.

Initialization: The traditional manner to initialize the estimates in any pursuit algorithm is to choose each action for the same fixed number of times to obtain a rough estimate of the reward probability of each action. This step is not really important and can be skipped and we rather can start from a reward estimate of 0.5 for each action in the absence of any priori information.

Initialization:

$t = 0$

For i= 1 to 2^K Do:

$u_{\{K,i\}}(0) = 1$

$v_{\{K,i\}}(0) = 2$

$\hat{d}_{\{K,i\}}(0) = \frac{u_{\{K,i\}}(0)}{v_{\{K,i\}}(0)}$

EndFor

Loop

1) $0 \leq k < K-1$: **Levels 0 to** $K-1$
   - LA $\mathcal{A}_{\{0,1\}}$ selects an action by randomly sampling according to the action probability vector $[p_{\{1,1\}}(t), p_{\{1,2\}}(t)]$.
   - Let $j_1(t)$ denote the index of the chosen action where $j_1(t) \in \{1,2\}$.
   - The next LA becomes consequently activated $\mathcal{A}_{\{1,j_1(t)\}}$ which in turn chooses an action according to its probability vector and thus activates the next LA at level '2'.
   - The procedure continues recursively until reaching an LA at level $K-1$.
   - Let $\mathcal{A}_{\{k,j_k(t)\}}$ denote the set of activated LA, where j_k denotes the activated LA at level k.
2) $k = K$: **Level** K
   - Update $\hat{D}_{\{K,j_K(t)\}}$ based on the response from the Environment at the leaf level, K:
   $$u_{\{K,j_K(t)\}}(t) = u_{\{K,j_K(t)\}}(t-1) + (1-R(t))$$
   $$v_{\{K,j_K(t)\}}(t) = v_{\{K,j_K(t)\}}(t-1) + 1$$
   $$\hat{d}_{\{K,j_K(t)\}}(t) = \frac{u_{\{K,j_K(t)\}}(t)}{v_{\{K,j_K(t)\}}(t)}.$$
   - For all other "leaf actions", where $j \in \{1,...,2^k\}$ and $j \neq j_K(t)$,
   $$u_{\{K,j\}}(t) = u_{\{K,j\}}(t-1)$$
   $$v_{\{K,j\}}(t) = v_{\{K,j\}}(t-1)$$
   $$\hat{d}_{\{K,j\}}(t) = \frac{u_{\{K,j\}}(t)}{v_{\{K,j\}}(t)}.$$
3) The reward estimates for all other actions along the path from root to the leaf node, $0 < k < K-1$ are defined according to a recursive procedure[7], where the LA at any one level inherits the feedback from the LA at the next level:
   $$\hat{d}_{\{k,j\}}(t) = \max\left(\hat{d}_{\{k+1,2j-1\}}(t), \hat{d}_{\{k+1,2j\}}(t)\right).$$
4) Proceed to updating the probability vectors in question along the path leading to the chosen action as follows:
   - By definition, each LA $j \in \{1,...,2^k\}$ at depth k, denoted by $\mathcal{A}_{\{k,j\}}$, where $0 \leq k \leq K-1$, possesses two actions $\alpha_{\{k+1,2j-1\}}$ and $\alpha_{\{k+1,2j\}}$. Let $j^h(t) \in \{2j-1, 2j\}$ be the biggest of the elements between $\hat{d}_{\{k+1,2j-1\}}(t)$ and $\hat{d}_{\{k+1,2j\}}(t)$.
   - Let $\overline{j^h(t)} = \{2j-1, 2j\} \setminus j^h(t)$ be the opposite action to $j^h(t)$, i.e., the other action that has the lower reward estimate among the two.
   - Update $p_{\{k,j^h(t)\}}$ and $p_{\{k,\overline{j^h(t)}\}}$ using the estimates $\hat{d}_{\{k+1,2j-1\}}(t)$ and $\hat{d}_{\{k+1,2j\}}(t)$ as:
   If $R(t) = 0$ **Then**
   $$p_{\{k,\overline{j^h(t)}\}}(t+1) = (1-\lambda) p_{\{k,\overline{j^h(t)}\}}$$
   $$p_{\{k,j^h(t)\}}(t+1) = 1 - p_{\{k,\overline{j^h(t)}\}}(t+1).$$
   Else
   $$p_{\{k,\overline{j^h(t)}\}}(t+1) = p_{\{k,\overline{j^h(t)}\}}(t)$$
   $$p_{\{k,j^h(t)\}}(t+1) = p_{\{k,j^h(t)\}}(t).$$
   EndIf
   - For each $\mathcal{A}_{\{k,j\}}$, we consider its probability vector and test if one of the action probabilities $p_{\{k+1,2j-1\}}$ and $p_{\{k+1,2j\}}$ overpasses a threshold T, where T is an arbitrarily close to unity strictly positive number, the probability vector for this LA is frozen in terms of updates, with the larger action probability rounded to unity.
5) $t = t+1$

EndLoop

End Algorithm HCPA

F. Remarks

Based on the above formulation of the problem and the solution that we have proposed, the following remarks are pertinent:

1) The reader will observe that we have used the CPA, and *its* operations at the lower level of the tree have been trickled up to discard the less optimal actions. One can then wonder if such a paradigm can be applicable to design LA solutions for a large number of actions

[7]To be more precise, the LA found at level $K-2$, integrates the feedback from the parent LA at level $K-1$ as:
$$\hat{d}_{\{K-2,j\}}(t) = \max\left(\hat{d}_{\{K-1,2j-1\}}(t), \hat{d}_{\{K-1,2j\}}(t)\right)$$
and so on. As a consequence, notice that *at every level*, the reward vector estimates of the actions of every LA, are composed of the respective *maxima* of the rewards of all the actions of the entire subtrees rooted at their children.

when the primitive machine is an FSSA, a VSSA (like the L_{R-I}), or any of the descritized LA like the (DL_{RI}). The answer to this question is in the negative. The reason for this is the following: If a two-action VSSA is used to distinguish between two actions at the leaves, the decision of the superior action can be trickled up *only after the LA has converged*. This implies that all the LA at any lower level must converge before the computations at a higher level can take place. Of course, this is because of the absence of the estimates of the corresponding reward probabilities.

2) The use of hierarchies of LA is not completely new. The most noteworthy example is the result of Baba *et al* in [5]. However, the latter paper does not utilize the pursuit concept, but rather the relative reward strength strategy proposed by Simha and Kurose in [48]. As one observes, we have advocated the coordinated use of the *estimates* of the reward probabilities *and* the action probability vector that have been used to design the pursuit families of LA, and which are far superior to merely using the action probability vectors, and/or the rewards and/or their relative strengths.
3) A consequence of the above remarks is that since we are using the estimates of the corresponding reward probabilities at each level, and trickling *them* up the tree, any other pursuit algorithm could have been used just as effectively as the CPA. Indeed, instead of the CPA, one could have used the DPA [27, 28, 36, 39, 63] or any of the members of the family of Bayesian pursuit algorithms [64, 67]. In the former case, the probability updates given in Step 4 of the algorithm will be done in a discretized manner. In the latter case, the estimates will not be obtained using a ML scheme as in Step 2, but rather using a Bayesian updating method.
4) We have all along assumed that the tree structure is a binary tree. However, we can easily extend the design to have a fixed number of children at each nodes, say Z. In that case, we will invoke a Z-action CPA or DPA at each level, and the corresponding proofs would involve the Z-action Hoeffding inequality instead of the 2-action Hoeffding inequality. The details of how this is done are rather trivial and thus omitted in the interest of brevity.
5) The story is not complete without mentioning that the algorithm above can be trivially parallelized. The details of such a parallelized solution are omitted.

The proof of convergence of the algorithm follows.

III. Proof of Convergence

To prove the convergence of the algorithm, we follow the same four-step method as in [65] and [66]. The reader will observe that the kernel of the proof of the algorithm follows the proofs used for the CPA. The finer details essentially deal with understanding that the proof now works level-by-level, and that the trickling up process is, indeed, effective by virtue of the properties of the "max" operator.

A. The Moderation Property of HCPA

The property of moderation can be described precisely by Theorem 1. This implies that under the HCPA, by utilizing a sufficiently small value for the learning parameter, λ, each action will be selected an arbitrarily large number of times.

Theorem 1: For any given constants $\delta > 0$ and $M < \infty$, there exist a positive learning parameter $\lambda_0 < 1$ and a time instant $t_0 < \infty$, such that under the HCPA algorithm, for all $\lambda < \lambda_0$,

$$Pr\{\text{All actions are selected at least } M \text{ times each before time } t_0\} > 1-\delta.$$

Proof: The way that we prove the moderation property for HCPA is similar to the proof of the corresponding moderation property for the CPA in [45].

Observe that there are 2^K actions at depth K, denoted as $\alpha_{\{K,j\}}$, where $j \in \{1,...,2^K\}$.

Let $Y^t_{\{K,j\}}$ be the number of times action $\alpha_{\{K,j\}}$ is chosen up to time t.

We want to prove $Pr\left(Y^t_{\{K,j\}} > M\right) \geq 1-\delta$, and this equivalent to proving $Pr\left(Y^t_{\{K,j\}} \leq M\right) \leq \delta$.

As the events $\left\{Y^t_{\{K,j\}} = l\right\}$ and $\left\{Y^t_{\{K,j\}} = n\right\}$ are mutually exclusive for $l \neq n$, we have:

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

$Pr\left(\alpha_{\{K,j_K\}} \text{is chosen}\right) = p_{\{K,j_K\}} p_{\{K-1,j_{K-1}\}} \cdots p_{\{0,j_0\}}$, where:

$j_{K-1} = \lceil j_K/2 \rceil$, $j_{K-2} = \lceil j_{K-1}/2 \rceil$, ..., and $j_0 = \lceil j_1/2 \rceil$.

$$\begin{aligned} &Pr\left(\alpha_{\{K,j_K\}} \text{is chosen at time } t\right) \\ &= p_{\{K,j_K\}}(t) p_{\{K-1,j_{K-1}\}}(t) \cdots p_{\{0,j_0\}}(t). \end{aligned}$$

To simplify arguments, we assume that all the LA have same value for the corresponding action probability vector, $P(0)$, at the beginning. Then:

$Pr\left(\alpha_{\{K,j_K\}} \text{is chosen at time } t\right) \geq p(0)^K (1-\lambda)^{tK}$

$Pr\left(\alpha_{\{K,j_K\}} \text{is not chosen at time } t\right) < (1-p(0)^K(1-\lambda)^{tK})$

The probability that action $\alpha_{\{K,j\}}$ is chosen at most M times among t choices has the upper bound:

$Pr\left(Y^t_{\{K,j\}} \leq M\right) = \sum_{l=1}^{M} Pr\left(Y^t_{\{K,j\}} = l\right) \leq \sum_{l=1}^{M} \binom{t}{l} (1)^l \psi^{t-l}$, where $\psi = \left(1-p(0)^K(1-\lambda)^{tK}\right)$. By definition, $0 < \lambda < 1$, therefore $\psi < 1$. We can also choose to avoid dependence in the number levels K of the tree: $(1-\lambda)^{tK} = p(0)$, then $\lambda = 1-p(0)^{-(tK)}$, and $\psi = 1-p(0)^K p(0) = 1-p(0)^{K+1}$.

As $\binom{t}{l} \leq t^l$, we have $Pr\left(Y^t_{\{K,j\}} \leq M\right) \leq \sum_{l=1}^{M} t^l \psi^{t-l} \leq M t^M \psi^{t-M}$.

When $t \to \infty$, $\lim_{t\to\infty} M t^M \psi^{t-M} = M \lim_{t\to\infty} \frac{t^M}{(1/\psi)^{t-M}}$, and by L'Hopital's rule,

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

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 $Y^t_{\{K,j\}} \geq M$. Therefore $Pr\left(Y^t_{\{K,j\}} \geq M\right) \geq Pr\left(Y^{t(j)}_{\{K,j\}} \geq M\right)$.

Therefore $Pr\left(Y^t_{\{K,j\}} \leq M\right) \leq \delta$ for all $t > t(j)$.

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 $Pr\left(Y^t_{\{K,j\}} \leq M\right) \leq \delta$. Theorem 1 is thus proven. □

B. Marginality at each level along the optimal path

Given that each action $\alpha_{\{K,j\}}$ will be selected a sufficiently large number of times, we now prove that along the optimal path, the reward estimate of the optimal action will remain the largest with a sufficiently large probability.

We denote $q_{\{k,j^*_k\}}$[8] as the probability that the reward estimate of the optimal action, $\hat{d}_{\{K,j^*_K\}}$, is the largest among all actions of the tree rooted at LA $\mathcal{A}_{\{k,j^*_k\}}$. More specifically:

- At the root level LA: $q_{\{0,0\}}$ is the probability that $\hat{d}_{\{K,j^*_K\}}$ is maximum among all the actions whose tree is rooted at the root LA $\mathcal{A}_{\{0,0\}}$. Note that there are 2^K actions that compete for having the best reward estimate.
- Second Level LA: $q_{\{1,j^*_1\}}$ is the probability that $\hat{d}_{\{K,j^*_K\}}$ is max among all actions of the tree rooted at LA $A_{\{1,j^*_1\}}$. Note that there are $2^{(K-1)}$ actions that compete for having the best reward estimate at this level.
- Interior Level LA: Without belaboring the point, we mention that the above statements are recursively true as one follows the path down the tree at every level.
- The last level LA: $q_{\{K-1,j^*_{K-1}\}}$ is the probability that $\hat{d}_{\{K,j^*_K\}}$ is maximum among the two actions at the last level, i.e., the maximum of the two actions of the LA $\mathcal{A}_{\{K-1,j^*_{K-1}\}}$. Thus, there are exactly 2 actions that compete for having the best reward estimate at this level.

Theorem 2: Given a $\delta \in (0,1)$, there exists a time instant $t_0 < \infty$, such that $\forall t > t_0$ and $\forall k \in \{0,1,...,K-1\}$:

$$q_{\{k,j^*_k\}} > 1-\delta.$$

Proof: To prove this result we first note that $q_{\{0,0\}} < q_{\{1,j*_1\}} < ... < q_{\{K-1,j^*_{K-1}\}}$, since the probability to be the best from among a set of actions is less than that from among a subset of the actions. Therefore, to prove Theorem 2, we only need to prove that $q_{\{0,0\}} > 1-\delta$. In other words, our goal is to prove that after a sufficiently large number of times, the reward estimate of the optimal action will remain the greatest among all actions with an arbitrarily large probability. Given the fact that Theorem 1 is proven, the assertion and proof for Theorem 2 become identical to the corresponding assertion and proof for the CPA given in [65] and [66]. To avoid unnecessary repetition, we omit the additional details of the proof. □

C. Submartingale Property

Theorem 3: Under the HCPA, the quantity $\left\{p_{\{k,j^*_k\}}(t)_{t>t_0}\right\}, k \in \{0,1,...,K-1\}$ is a submartingale.

[8]To render the notation less cumbersome, we use the symbol $*$ to mark the index along the path.

Proof: To formalize prove this assertion, we first explicitly calculate $E\left[p_{\{k,j^*_i k\}}(t)\right]$. Using the HCPA's updating rule, we have Eq (1):

$$\begin{aligned}
&E\left[p_{\{k,j^*_k\}}(t+1)|P(t)\right] \\
=&\sum_j p_{\{k,j\}}\left(d_j\left(q_{\{k,j^*_k\}}\left[(1-\lambda)p_{\{k,j^*_k\}}+\lambda\right]\right.\right. \\
&\left.\left.+\left(1-q_{\{k,j^*_k\}}\right)\left[(1-\lambda)p_{\{k,j^*_k\}}\right]\right)+(1-d_j)p_{\{k,j^*_k\}}\right) \\
=&p_{\{k,j^*_k\}}+\lambda\left(q_{\{k,j^*_k\}}-p_{\{k,j^*_k\}}\right)\sum_{j=1,2}p_{\{k,j\}}d_j, \qquad (1)
\end{aligned}$$

where, in the interest of conciseness, we omit the reference to time and represent $p_{\{k,j^*_k\}}(t)$ and $q_{\{k,j^*_k\}}(t)$ as $p_{\{k,j^*_k\}}$ and $q_{\{k,j^*_k\}}$ respectively. Thus,

$$\begin{aligned}
Diff_{p_{\{k,j^*_k\}}(t)} &= E\left[p_{\{k,j^*_k\}}(t+1)|P(t)\right]-p_{\{k,j^*_k\}}(t) \\
&= \lambda\left(q_{\{k,j^*_k\}}-p_{\{k,j^*_k\}}\right)\sum_{j=1,2}p_{\{k,j\}}d_j.
\end{aligned}$$

Invoking the definition of a submartingale, we know that if for all $t > t_0$, we have $Diff_{p_{\{k,j^*_k\}}(t)}$, i.e., $q_{\{k,j^*_k\}}(t)-p_{\{k,j^*_k\}}(t) > 0$, then $\left\{p_{\{k,j^*_k\}}(t)_{t>t_0}\right\}$ is a submartingale. We now invoke the terminating condition for the HCPA, in which we force the learning process to jump to the absorbing state and attain convergence if $p_{\{k,j\}}(t) > T = 1-\varepsilon, (k \in \{0,1,...,K-1\}, j \in \{1,2\})$. Therefore, if we set the quantity $(1-\delta)$ defined in Theorem 2 to be greater than the threshold T, then as per Theorem 2, there exists a time instant $t_0 < \infty$, such that for every single time instant subsequent to $t > t_0$, and for all $k \in \{0,1,...,K-1\}$, $q_{\{k,j^*_k\}}(t) > 1-\delta > T > p_{\{k,j^*_k\}}(t)$, which, in turn, guarantees that $\{p_{\{k,j^*_k\}}(t)_{t>t_0}\}$ is a submartingale. Hence the theorem! □

We can now finally prove the ε-optimality of the HCPA.

D. $Pr\left(p_{\{k,j^*_k\}}(\infty)=1\right) \to 1$ under the HCPA

When the action probabilities of selecting the optimal action along the path, $p_{\{k,j^*_k\}}$, converges, the HCPA converges to the optimal action with a probability that is arbitrarily close to unity. We shall formally assert and prove this.

Theorem 4:

The HCPA is ε-optimal in all random Environments. More formally, let $T = 1-\varepsilon$ be a value arbitrarily close to 1, with ε being arbitrarily small. Then, given any $1-\delta > T$, there exists a positive integer $\lambda_0 < 1$ and a time instant $t_0 < \infty$, such that for all learning parameters $\lambda < \lambda_0$ and for all $t > t_0$, $q_{\{k,j^*_k\}}(t) > 1-\delta$, and the quantity $Pr\left(p_{\{k,j^*_k\}}(\infty)=1\right) \to 1$, where $k \in \{0,1,...,K-1\}$.

Proof: We prove Theorem 4 level-wise. Firstly consider the case when $k = 0$, in which case $j_0 = 0$ or 1. We denote the index of the optimal action at this level as m, and then prove that $Pr\left(p_{\{0,m\}}(\infty)=1\right) \to 1$.

According to Theorem 3, $\{p_{\{0,m\}}(t)_{t>t_0}\}$ is a submartingale, and thus, invoking the submartingale convergence theory [32]:

$$p_{\{0,m\}}(\infty) = 0 \text{ or } 1.$$

If we denote e_j as the unit vector with the j^{th} element being 1, then $p_{\{0,m\}}(\infty) = 1$ is equivalent to the assertion that $P_{\{0\}}(\infty) = e_m$. If we define the convergence probability

$$\Gamma_m(P) = Pr\left(P_{\{0\}}(\infty) = e_m | P_{\{0\}}(0) = P\right),$$

our task is to now prove:

$$\Gamma_m(P) \to 1. \tag{2}$$

Similar to the proof in [65] and [66], $\Gamma_m(P)$ can be observed by investigating a Regular function of P. As before, the goal is to find a proper *Subregular* function of P, denoted as $\Phi(P)$, which also satisfies the boundary conditions $\Phi(e_m) = 1$ and $\Phi(e_j) = 0, (\text{for } j \neq m)$, which will then guarantee to bound $\Gamma_m(P)$ from below.

Let $\Phi(P)$ as a function of P. We now define an operator U as

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

We now repeatedly apply U to get the result of the t-step invocation of U as:

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

Consider a specific instantiation of Φ to be the function Φ_m, defined below as:

$$\Phi_m(P) = e^{-x_m p_{\{0,m\}}},$$

where x_m is a positive constant. Then, under the HCPA,

$$\begin{aligned}
&U(\Phi_m(P)) - \Phi_m(P) \\
&= E[\Phi_m(P(t+1))|P(t) = P] - \Phi_m(P) \\
&= E\left[e^{-x_m p_{\{0,m\}}(t+1)}|P(t) = P\right] - e^{-x_m p_{\{0,m\}}} \\
&= \sum_{j=1,2} e^{-x_m\left((1-\lambda)p_{\{0,m\}}+\lambda\right)} p_{\{0,j\}} d_j q_{\{0,0\}} \\
&\quad + \sum_{j=1,2} e^{-x_m\left((1-\lambda)p_{\{0,m\}}\right)} p_{\{0,j\}} d_j \left(1 - q_{\{0,0\}}\right) \\
&\quad + \sum_{j=1,2} e^{-x_m p_{\{0,m\}}} p_{\{0,j\}}(1 - d_j) - e^{-x_m p_{\{0,m\}}} \\
&= \sum_{j=1,2} p_{\{0,j\}} d_j e^{-x_m p_{\{0,m\}}} \left(q_{\{0,0\}} e^{-x_m\left(1-p_{\{0,m\}}\right)\lambda}\right. \\
&\quad \left. + \left(1 - q_{\{0,0\}}\right) e^{x_m p_{\{0,m\}}\lambda} - 1\right).
\end{aligned}$$

Our task is to determine a proper value for x_m such that $\Phi_m(P)$ is superregular, i.e.,

$$U(\Phi_m(P)) - \Phi_m(P) \leq 0.$$

This is equivalent to solving the following inequality:

$$q_{\{0,0\}} e^{-x_m\left(1-p_{\{0,m\}}\right)\lambda} + \left(1 - q_{\{0,0\}}\right) e^{x_m p_{\{0,m\}}\lambda} - 1 \leq 0, \tag{3}$$

which, is equivalent to

$$x_m \left(x_m - \frac{2\left(q_{\{0,0\}}\left(1-p_{\{0,m\}}\right) + p_{\{0,m\}}\left(1-q_{\{0,0\}}\right)\right)}{\lambda\left(q_{\{0,0\}} - 2q_{\{0,0\}}p_{\{0,m\}} + p^2_{\{0,m\}}\right)}\right) \leq 0.$$

As x_m is defined as a positive constant, we have

$$0 < x_m \leq \frac{2\left(q_{\{0,0\}}\left(1-p_{\{0,m\}}\right) + p_{\{0,m\}}\left(1-q_{\{0,0\}}\right)\right)}{\lambda\left(q_{\{0,0\}} - 2q_{\{0,0\}}p_{\{0,m\}} + p^2_{\{0,m\}}\right)}. \tag{4}$$

If we denote

$$x_{m_0} = \frac{2\left(q_{\{0,0\}}\left(1-p_{\{0,m\}}\right) + p_{\{0,m\}}\left(1-q_{\{0,0\}}\right)\right)}{\lambda\left(q_{\{0,0\}} - 2q_{\{0,0\}}p_{\{0,m\}} + p^2_{\{0,m\}}\right)},$$

we have $x_{m_0} > 0$, implying that when $\lambda \to 0$, $x_{m_0} \to \infty$.

To satisfy the bounday conditions, we now introduce another function

$$\phi_m(P) = \frac{1 - e^{-x_m p_{\{0,m\}}}}{1 - e^{-x_m}},$$

where x_m is the same as defined in $\Phi_m(P)$. As per the property that if $\Phi_m(P) = e^{-x_m p_m}$ is a superregular (subregular), then $\phi_m(P) = \frac{1-e^{-x_m p_m}}{1-e^{-x_m}}$ is a subregular (superregular) [32], the quantity x_m, as defined in Eq. (4), which renders $\Phi_m(P)$ to be superregular, makes the $\phi_m(P)$ to be subregular.

Obviously, $\phi_m(P)$ satisifes the boundary conditions, i.e.,

$$\phi_m(P) = \frac{1 - e^{-x_m p_{\{0,m\}}}}{1 - e^{-x_m}} = \begin{cases} 1, & \text{when } P = e_m, \\ 0, & \text{when } P = e_j. \end{cases}$$

Therefore, as per the property of Regular functions,

$$\Gamma_m(P) \geq \phi_m(P) = \frac{1 - e^{-x_m p_{\{0,m\}}}}{1 - e^{-x_m}}. \tag{5}$$

As Eq. (5) holds for every x_m bounded by Eq. (4), we take the greatest value x_{m_0}. Moreover, as $\lambda \to 0$, $x_{m_0} \to \infty$, whence $\Gamma_m(P) \to 1$. We thus shown that $Pr\left(p_{\{0,m\}}(\infty) = 1\right) \to 1$.

With little imagination, the reader will observe that the same proof methodology can be applied for cases where $k = 1, 2, .., K-1$, thus proving Theorem 4. □

Having established the theoretical properties of the HCPA we shall now demonstrate its power by simulating and comparing it with the benchmark reported LA.

IV. Experimental Results

In order to assess the performance of the LA-based schemes, we have conducted thorough experiments involving different "large" sets of actions. The main finding that we would like to crystallize is that the traditional VSSA are inferior to our hierarchical solutions both in terms of speed and accuracy. The reason behind this as explained throughout the article is that as the number of actions becomes lager, many of the actions will suffer from the fact that their action probabilities are small and thus they will be chosen quite rarely. In this manner, a direct usage of VSSA in the context of estimator LA will imply that each action should be chosen for a very large number of trials. In addition, the estimates will be consequently inaccurate. Our HCPA remediates to those two aforementioned issues. In the simulation, we have focused on two key performance metrics: the accuracy of convergence and the speed of the convergence. We use those two key performance criteria in the comparisons against legacy LA solutions.

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

TABLE I: In this table we provide the reward probabilities of the 64 actions used in our experiments. For the case of 16 and 32 actions-environments, the reward probabilities correspond to the 16 and 32 first entries in the table, respectively.

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

A. The Data Sets for the Environment

In the field of LA, the adopted datasets in the literature are usually characterized by a number of actions not exceeding ten actions at most. When it comes to higher number of actions, there is no existing established benchmark and therefore we propose a benchmark that can be adopted by other researchers. To make our problem far from being trivial, the total numbers of actions was fixed to be 16, 32 and 64. Once the number of actions was set, the reward probabilities associated with the different actions are drawn uniformly from the unit interval. Obviously, the higher the number of actions, the more difficult is the Environment. The first 16 and 32 elements in Table I give the reward probabilities for an Environment with 16 and 32 actions respectively. The whole 64 elements of the Table I give the reward probabilities for an Environment consisting of 64 actions.

B. Convergence of the HCPA Algorithm

As the convergence of the HCPA algorithm has already been formally proven in Section III, our task in this sub-section, is to validate this proof through simulations.

From the mathematical proof, one understands that if λ is sufficiently small, we are guaranteed that the HCPA will converge to the action with the maximum reward probability with a probability increasingly close to 1. We define the optimal λ value as the maximum λ value for which the LA to *consistently* converge to the optimal action, i.e., the action with highest reward. Intuitively, the optimal λ depends on the configurations for the Environment. We started decreasing the value λ until we found a value that consistently yields convergence for 400 consecutive runs to the optimal action. The obtained λ for which this takes place is denoted the optimal λ. From our experimental tests, for 64 actions, the optimal λ was found to be 0.000051. For 32 and 16 actions, we found that the the optimal values for λ are respectively 0.00085 and 0.0065. We notice that the optimal λ tends to increase as the Environment becomes more challenging.

C. Average Convergence Iterations

We investigate in this experiment the average number of iterations required for accomplishing convergence [9] in Tables II. Furthermore, we report the standard deviation of the iterations required for convergence. We compare the results of the HCPA with the legacy LA schemes namely L_{R-I} and CPA. The optimal λ values utilized for the HCPA in this experiment focusing on average convergerence time are the ones reported in Section IV-B while we use the same procedure for obtaining the optimal λ values for both CPA and L_{R-I} based on the same approach explained in Section IV-B.

We reckon that the HCPA has converged when all the LA along the optimal path leading to the correct leaf node have converged to an action probability larger than 0.99. In a similar manner, we deem that the the CPA and the L_{R-I} have converged correctly if the LA had converged to the correct action with an action probability larger than 0.99. The results in the table represent the average of an ensemble of 400 independent replications using the optimal λ obtained from the previous experiment. As we can be see from Table II, the HCPA yields superior performance to CPA and L_{R-I}. This superiority becomes more clear as the number of actions increases. For instance, for the 64-action environment,the HCPA required 115,295 iterations in average while L_{R-I} required 644,234 iterations. In other terms, the HCPA was able to converge in less than 18% of the corresponding time for the L_{R-I}. Since those results are recurrent and general we should not detail more those numbers here. Such numbers clearly confirm the efficiency of the hierarchical structure of the HCPA as the number of actions becomes larger.

D. Convergence for 128 Actions

The premise of this paper was to devise an LA-based scheme that could be effective when the number of actions was, indeed, large. To clearly demonstrate the power of the HCPA for an even larger number of actions than those reported in the previous subsections, we also tested it for two environments with 128 actions (and in the next sub-section we report for one environment with 256 actions). To the best of our knowledge, extremely sparse results are reported in the field of LA for large environments and specially when it comes to results of experimental nature, and therefore, we consider the experiments and the scheme reported here as truly quite ground-breaking.

We first report the results for which the HCPA was tested in two environments with 128 actions. In both the cases, the reward probabilities were randomly chosen in the interval

[9]We conducted different experiments for different randomly-generated environments. However, due to space limitations, we only report the results for a single settings that is a representative for other settings.

TABLE II: The simulation results obtained for various environments with different numbers of actions.

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

$[0,1]$. In these cases, unlike the environments in the previous sub-section, instead of listing the reward probabilities, we have opted to an alternative representation by depicting them along the line in Figures 2 and 3 respectively.

For the first environment illustrated by Figure 2, the L_{R-I} converged in 734,474 iterations in average while the CPA achieved convergence within 543,529 steps, which means 26% less time than the L_{R-I}. Amazingly, the HCPA only needed 266,257 steps in average over 400 experiments for convergence. This makes the HCPA 51% faster than CPA and almost 64% faster than the L_{R-I}. The results are representative and confirm the superiority of the HCPA over legacy LA schemes.

In the case of the second environment plotted in Figure 3, the L_{R-I} required 3,760,704 steps for absolute convergence for an ensemble of 400 trials. This, in and of itself, shows that this learning problem was much more difficult than the one displayed in Figure 2. In this case, the CPA required 682,853 steps - which represented a decrease of about 81%. Astonishingly, the HCPA needed only 476,511 steps. This is equivalent to an advantage of about 30% over the CPA and more than 87% over the L_{R-I}! The advantage of the current HCPA cannot be disputed.

E. Convergence for 256 Actions

The reason why we have, in the first instance, only gone for a maximum of 128 actions is because while the newly introduced LA converge "relatively" quickly, the traditional LA converge very slowly, both in terms of the number of iterations and the number of computations needed per iteration. Running an ensemble of experiments for environments with even larger numbers of actions, would takes weeks of computational power. However, in the interest of completeness, we have also included the results for an environment with 256 actions. Again, rather than list the reward probabilities, we have plotted them in Figures 4. In this case, of the many environments that we considered, we selected a case where the "distance" in the probability space between the largest reward probability and the second largest reward probability was relatively large. This rendered the learning environment to be much easier than the ones encountered in the Section IV-D, permitting the computations to be achieved in a reasonable time-frame. We report the consequence of this choice presently.

For this 256-action environment, the L_{R-I} required 1,196,918 steps for absolute convergence for an ensemble of 400 trials. In this case, the CPA required 542,042 steps - which represented a decrease of about 55%. However, the HCPA needed only 253,739 steps. Clearly, the HCPA is the winner. This is equivalent to an advantage of about 57% over the CPA and more than 79% over the L_{R-I}. The fact that we selected a case where the "distance" in the probability space between the largest reward probability and the second largest reward probability was relatively large, explains why the HCPA with 256 actions required less number of steps (253,739 than in the first 128 case, where we needed 266,257 steps), because the 128 case represented a more challenging learning task.

F. Discussions

There are also conceptual differences between taking estimates and using them in the Pursuit paradigm, and directly making decisions based on these estimates[10]. First of all, the power of the LA-paradigm precisely works with the understanding that one does not make decisions solely based on the estimates of the reward probabilities. This is because, we would like to choose the actions *even while we do the estimation*, and this is achieved by simply maintaining the additional (linear space and time) Action Probability Vector. This, in turn, means that we can minimize the number of times that the inferior actions are chosen for the estimation phase. In a typical Pursuit scheme, some of the very inferior actions will scarcely be chosen subsequent to a few iterations after the initialization stage. Rather, the learning scheme spends more effort in taking more accurate estimates for the superior, competitive actions. This makes the algorithm converge much more rapidly, as reported by the LA community.

One additional difference between our method and one that makes decisions after merely taking estimates of each action, say 2,000 times, is that the latter would rarely guarantee a 100% convergence over the entire ensemble. This is exactly our experience in these Environments. When we use the configuration of Figure 3 with 128 actions, the HCPA required 226,257 steps to converge, which is approximately 2,080 trials for each action. However, when we attempt to estimate for each action, with the same configuration 2,080 times the system does not converge with 100% accuracy to the correct action (i.e., to the one which has the maximum reward probability). It is very interesting to observe that if we enforce the condition to have 100% convergence to the best action, the number of estimates required per action is around 3,000! This demonstrates why researchers have opted to use the LA paradigm over the straightforward estimation methods for over four decades!

When we discuss the actual simulations, it is also very important that the following is highlighted. The reason behind this seemingly slow convergence is the way by which we have done the simulation. In fact, we have chosen a value of λ that gives us a consistent convergence[11] to the optimal

[10] We are grateful for the Referee who requested this explanation.

[11] Obviously, we could have determined the optimal tuning parameter that guaranteed convergence with a certain confidence, e.g., 99% of the time.

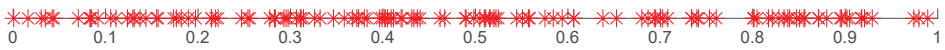

Fig. 2: The first 128-action Environment.

Fig. 3: The second 128-action Environment.

Fig. 4: The 256-action Environment that we have studied.

action over the entire ensemble of experiments. We can refer to such a procedure as a "Fine Tuning Procedure for Consistent Convergence". The reason for us working with such a "Fine Tuning Procedure for Consistent Convergence" philosophy is that it boasts a more fair comparison of the the different LA algorithms. In fact, in order to compare two LA algorithms we need to consider *both* speed of convergence and accuracy. Our comparison methodology tries to compare the speed of the LA algorithms when their accuracy is fixed, i.e., after obtaining the corresponding tuning parameters to obtain consistent convergence. Observe that the "Fine Tuning Procedure for Consistent Convergence" led us to very low values of λ – as small as 0.000051 for the case of 64 actions - which is much smaller that the smallest values for λ commonly used in the literature (usually around 0.001). By adopting such a comparison mechanism, we can guarantee that the comparisons are done on a level field, where we can obtain the average number of iterations needed for all the schemes tested to yield the same accuracy.

Finally, the distribution of the reward probabilities around the optimal actions plays a crucial rule in the convergence. In fact, whenever there are competing actions to the optimal action, i.e., actions that have a reward probability close to the optimal action reward probability, the convergence time increases significantly due to two reasons. The first reason is that our hierarchical pursuit LA needs more time to distinguish between the actions by their estimates. Secondly, the "Fine Tuning Procedure for Consistent Convergence" leads to even lower values for λ when compared to a simpler environment with well-separated optimal and second sub-optimal actions.

V. Conclusions

In this paper, we laid the foundations of a new model for devising efficient and rapid Learning Automata (LA) schemes specially adequate for a large number of actions. The settings in which the number of actions is large is particularly challenging since the dimensionality of the probability vector becomes consequently large and many of its components tend to decay in few iterations to small values under what the machine accuracy can permit *leading to the fact that they cease to be selected*. In this case, the LA will be inaccurate and the theoretical assumption that each action will be probed for a large number of times will not be fulfilled in practice if we use the family of estimator LA. In this paper, we introduce a novel paradigm that extends the Continuous Pursuit Algorithm's (CPA's) to the large set of actions. The most distinguishing characteristic of our scheme is the fact that is hierarchical and all the actions reside in the leave nodes. Further, at each level of the hierarchy we only need a *two*-action LA and thus we can easily eliminate the problem of having extremely low action probability. By design, all the LA of the hierarchy resort to the pursuit paradigm, and therefore the optimal action of each level trickles up towards the root. Thus, by recursively applying the "max" operator, in which, the maximum of several local maxima is a global maximum, the overall hierarchy converges to the optimal action. We also provide sound theoretical results that demonstrated that our scheme has ε-optimal convergence. Furthermore, we report comprehensive experiments with large set of actions namely 128 and 256 actions that demonstrate the power of our schemes both in terms of accuracy and convergence speed.

While the present paper has used the CPA as its kernel, we emphasize that the results presented here can, rather trivially, be extended for the families of discretized and Bayesian pursuit LA too. Indeed, as far as we know, there are no comparable results in the field of LA, in which the set of LA are superimposed on a tree-like structure and where each LA works with a pair of actions, rendering the action probability vector to be of dimension *two*, and the estimates to also always involve a *pair* of actions.

There are also algorithms that are based on Bayesian reasoning and Thompson Sampling [21, 22]. By adopting the concept of conjugate priors, these authors have shown that invoking Bayesian reasoning within the field of LA becomes computationally tractable. This concept has, thus, been used to construct the Bayesian LA (BLA). The question of cascading BLAs in a hierarchy, as we have done here, remains open.

There is also a significant difference between LA-based methods and those that are generally considered as Subspace Learning and Deep Learning methods. However, when the number of dimensions in the feature domain is large, we believe that the feature indices can be used as actions and that one can learn the best dimensions by invoking hierarchical LA like the HCPA to learn the best features. This will, certainly,

prove to be a valuable area for future research[12].

With regard to future work. we also note that, in this paper, we have not taken advantage of the concept of discretizing the probability space. This option would lead to even faster hierarchical CPA, inasmuch as discretized LA are known to yield the fastest LA schemes. The speed of the *Discretized* HCPA would be even faster than the HCPA presented here.

REFERENCES

[1] M. Agache and B. J. Oommen. Generalized pursuit learning schemes: new families of continuous and discretized learning automata. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 32(6):738–749, 2002.

[2] A. F. Atlassis, N. H. Loukas, and A. V. Vasilakos. The use of learning algorithms in ATM networks call admission control problem: A methodology. *Computer Networks*, 34:341–353, 2000.

[3] A. F. Atlassis and A. V. Vasilakos. The use of reinforcement learning algorithms in traffic control of high speed networks. *Advances in Computational Intelligence and Learning*, pages 353–369, 2002.

[4] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The non-stochastic multi-armed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[5] N. Baba and Y. Mogami. A new learning algorithm for the hierarchical structure learning automata operating in the nonstationary S-model random environment. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, 32(6):750–758, 2002.

[6] A. Baddeley and R. Turner. Spatstat: An R package for analyzing spatial point patterns. *Journal of Statistical Software*, 12:1–42, 2005.

[7] M. Barzohar and D. B. Cooper. Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:707–722, 1996.

[8] C. Bettstetter, H. Hartenstein, and X. Pï¿½rez-Costa. Stochastic properties of the random waypoint mobility model. *Journal Wireless Networks*, 10:555–567, 2004.

[9] M. L. Brandeau and S. S. Chiu. An overview of representative problems in location research. *Management Science*, 35:645–674, 1989.

[10] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári. X-armed bandits. *Journal of Machine Learning Research*, 12(May):1655–1695, 2011.

[11] N. Cesa-Bianchi and P. Fischer. Finite-time regret bounds for the multiarmed bandit problem. In *ICML1998*, pages 100–108, Madison, Wisconsin USA, Jul. 1998.

[12] J. J. Collins, C. C. Chow, and T. T. Imhoff. Aperiodic stochastic resonance in excitable systems. *Physical Review E*, 52:R3321–R3324, 1995.

[13] R. L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5:51–72, 1986.

[12] We are grateful for the anonymous Referee who suggested this.

[14] P.-A. Coquelin and R. Munos. Bandit algorithms for tree search. *arXiv preprint cs/0703062*, 2007.

[15] A. Couëtoux, J.-B. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard. Continuous upper confidence trees. In *International Conference on Learning and Intelligent Optimization*, pages 433–445. Springer, 2011.

[16] J. P. Cusumano and B. W. Kimble. A stochastic interrogation method for experimental measurements of global dynamics and basin evolution: Application to a two-well oscillator. *Nonlinear Dynamics*, 8:213–235, 1995.

[17] E. Even-Dar, S. Mannor, and Y. Mansour. Pac bounds for multi-armed bandit and markov decision processes. In *International Conference on Computational Learning Theory*, pages 255–270. Springer, 2002.

[18] S. Gelly and D. Silver. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 175(11):1856–1875, 2011.

[19] J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41(2):148–177, 1979.

[20] J. C. Gittins and D. M. Jones. A dynamic allocation index for the discounted multiarmed bandit problem. *Biometrika*, 66(3):561–565, 1979.

[21] O.-C. Granmo. Solving two-armed bernoulli bandit problems using a Bayesian learning automaton. *International Journal of Intelligent Computing and Cybernetics*, 3(2):207–234, 2010.

[22] O.-C. Granmo and S. Berg. Solving non-stationary bandit problems by random sampling from sibling kalman filters. In *Proceedings of IEA-AIE 2010*, pages 199–208, Cordoba, Spain, Jul. 2010.

[23] O.-C. Granmo and B. J. Oommen. Solving stochastic nonlinear resource allocation problems using a hierarchy of twofold resource allocation automata. *IEEE Transactions on Computers*, 59:545–560, 2009.

[24] J. Kabudian, M. R. Meybodi, and M. M. Homayounpour. Applying continuous action reinforcement learning automata (CARLA) to global training of hidden markov models. In *Proceedings of the International Conference on Information Technology: Coding and Computing , ITCC'04*, pages 638–642, Las Vegas, Nevada, 2004.

[25] S. Lakshmivarahan. *Learning Algorithms Theory and Applications*. New York Springer-Verlag, 1981.

[26] S. Lakshmivarahan and M. A. L. Thathachar. Absolutely expedient algorithms for stochastic automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 3:281–286, 1973.

[27] J. K. Lanctot and B. J. Oommen. On discretizing estimator-based learning algorithms. *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, 2:1417–1422, 1991.

[28] J. K. Lanctot and B. J. Oommen. Discretized estimator learning automata. *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, 22(6):1473–1483, 1992.

[29] M. R. Meybodi and H. Beigy. New learning automata based algorithms for adaptation of backpropagation algorithm pararmeters. *International Journal of Neural*

Systems, 12:45–67, 2002.

[30] S. Misra and B. J. Oommen. GPSPA: A new adaptive algorithm for maintaining shortest path routing trees in stochastic networks. *International Journal of Communication Systems*, 17:963–984, 2004.

[31] K. Najim and A. S. Poznyak. *Learning Automata: Theory and Applications*. Pergamon Press, Oxford, 1994.

[32] K. S. Narendra and M. A. L. Thathachar. *Learning Automata: An Introduction*. Prentice Hall, 1989.

[33] M. S. Obaidat, G. I. Papadimitriou, and A. S. Pomportsis. Learning automata: Theory, paradigms, and applications. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 32(6):706–709, December 2002.

[34] M. S. Obaidat, G. I. Papadimitriou, A. S. Pomportsis, and H. S. Laskaridis. Learning automata-based bus arbitration for shared-edium ATM switches. *IEEE Transactions on Systems, Man, and Cybernetics: Part B*, 32:815–820, 2002.

[35] B. J. Oommen. Absorbing and ergodic discretized two-action learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 16:282–296, 1986.

[36] 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*, 31(3):277–287, 2001.

[37] B. J. Oommen and T. D. S. Croix. Graph partitioning using learning automata. *IEEE Transactions on Computers*, 45:195–208, 1996.

[38] B. J. Oommen and T. D. S. Croix. String taxonomy using learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 27:354–365, Apr. 1997.

[39] B. J. Oommen and J. K. Lanctot. Discretized pursuit learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 20:931–938, 1990.

[40] M. Paola. Digital simulation of wind field velocity. *Journal of Wind Engineering and Industrial Aerodynamics*, 74-76:91–109, 1998.

[41] G. Papadimitriou. Hierarchical discretized pursuit nonlinear learning automata with rapid convergence and high accuracy. *IEEE Transactions on Knowledge and Data Engineering*, 6:654–659, 1994.

[42] G. I. Papadimitriou. Hierarchical discretized pursuit nonlinear learning automata with rapid convergence and high accuracy. *IEEE Transactions on Knowledge and Data Engineering*, 6(4):654–659, 1994.

[43] G. I. Papadimitriou and A. S. Pomportsis. Learning-automata-based TDMA protocols for broadcast communication systems with bursty traffic. *IEEE Communication Letters*, pages 107–109, 2000.

[44] A. S. Poznyak and K. Najim. *Learning Automata and Stochastic Optimization*. Springer-Verlag, Berlin, 1997.

[45] K. Rajaraman and P. S. Sastry. Finite time analysis of the pursuit algorithm for learning automata. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26:590–598, 1996.

[46] B. S. Rowlingson and P. J. Diggle. *SPLANCS: Spatial Point Pattern Analysis Code in S-Plus*. University of Lancaster, North West Regional Research Laboratory, 1991.

[47] F. Seredynski. Distributed scheduling using simple learning machines. *European Journal of Operational Research*, 107:401–413, 1998.

[48] R. Simha and J. F. Kurose. Relative reward strength algorithms for learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 19:388–398, 1989.

[49] M. A. L. Thathacha and P. S. Sastry. *Networks of Learning Automata: Techniques for Online Stochastic Optimization*. Kluwer Academic Publishers, 2004.

[50] M. A. L. Thathachar and B. J. Oommen. Discretized reward-inaction learning automata. *Journal of Cybernetics and Information Science*, pages 24–29, 1979.

[51] M. A. L. Thathachar and K. R. Ramakrishnan. A hierarchical system of learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 11:236–241, 1981.

[52] M. A. L. Thathachar and P. S. Sastry. Estimator algorithms for learning automata. In *Proceedings of the Platinum Jubilee Conference on Systems and Signal Processing*, pages 29–32, Bangalore, India, Dec. 1986.

[53] M. Tokic. Adaptive ε-greedy exploration in reinforcement learning based on value differences. In *KI'10 Proceedings of the 33rd annual German conference on Advances in artificial intelligence*, Karlsruhe, Germany, Sep. 2010. Springer.

[54] M. L. Tsetlin. Finite automata and the modeling of the simplest forms of behavior. *Uspekhi Matem Nauk*, 8:1–26, 1963.

[55] C. Unsal, P. Kachroo, and J. S. Bay. Simulation study of multiple intelligent vehicle control using stochastic learning automata. *Transactions of the Society for Computer Simulation International*, 14:193–210, 1997.

[56] C. Unsal, P. Kachroo, and J. S. Bay. Multiple stochastic learning automata for vehicle path control in an automated highway system. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 29:120–128, 1999.

[57] A. V. Vasilakos, M. P. Saltouros, A. F. Atlassis, and W. Pedrycz. Optimizing QoS routing in hierarchical ATM networks using computational intelligence techniques. *IEEE Transactions on Systems, Man and Cybernetics: Part C*, 33:297–312, 2003.

[58] C. J. C. H. Watkins. Learning from delayed rewards. *Ph.D. thesis. Cambridge University*, 1989.

[59] P. Whittle. Multi-armed bandits and the gittins index. *Journal of the Royal Statistical Society. Series B (Methodological)*, 42(2):143–149, 1980.

[60] A. Yazidi, O.-C. Granmo, and B. J. Oommen. Service selection in stochastic environments: A learning-automaton based solution. *Applied Intelligence*, 36:617–637, 2012.

[61] A. Yazidi, O.-C. Granmo, B. J. Oommen, and M. Goodwin. A novel strategy for solving the stochastic point location problem using a hierarchical searching scheme. *IEEE transactions on cybernetics*, 44(11):2202–2220, 2014.

[62] X. Zhang, O.-C. Granmo, and B. J. Oommen. The Bayesian pursuit algorithm: A new family of estimator learning automata. In *Proceedings of IEA-AIE 2011*,

pages 608–620, New York, USA, Jun. 2011. Springer.

[63] X. Zhang, O.-C. Granmo, and B. J. Oommen. Discretized Bayesian pursuit - a new scheme for reinforcement learning. In *Proceedings of IEA-AIE 2012*, pages 784–793, Dalian, China, Jun. 2012.

[64] 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*, 39:782–792, 2013.

[65] X. Zhang, O.-C. Granmo, B. J. Oommen, and L. Jiao. On using the theory of regular functions to prove the ε-optimality of the continuous pursuit learning automaton. In *Proceedings of IEA-AIE 2013*, pages 262–271, Amsterdan, Holland, Jun. 2013. Springer.

[66] X. Zhang, O.-C. Granmo, B. J. Oommen, and L. Jiao. A formal proof of the ε-optimality of absorbing continuous pursuit algorithms using the theory of regular functions. *Applied Intelligence*, 41(3):974–985, 2014.

[67] 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*, 20(3):797–808, Aug 2017.