A Novel Strategy for Solving the Stochastic Point Location Problem Using a Hierarchical Searching Scheme

Anis Yazidi, Ole-Christoffer Granmo, B. John Oommen, *Fellow, IEEE,* and Morten Goodwin

***Abstract*—Stochastic point location (SPL) deals with the problem of a learning mechanism (LM) determining the optimal point on the line when the only input it receives are stochastic signals about the direction in which it should move. One can differentiate the SPL from the traditional class of optimization problems by the fact that the former considers the case where the directional information, for example, as inferred from an Oracle (which possibly computes the derivatives), suffices to achieve the optimization—without actually explicitly computing any derivatives. The SPL can be described in terms of a LM (algorithm) attempting to locate a point on a line. The LM interacts with a random environment which essentially informs it, possibly erroneously, if the unknown parameter is on the left or the right of a given point. Given a current estimate of the optimal solution, all the reported solutions to this problem effectively move along the line to yield updated estimates which are in the neighborhood of the current solution.[1] This paper proposes a dramatically distinct strategy, namely, that of partitioning the line in a hierarchical tree-like manner, and of moving to relatively distant points, as characterized by those along the path of the tree. We are thus attempting to merge the rich fields of stochastic optimization and data structures. Indeed, as in the original discretized solution to the SPL, in one sense, our solution utilizes the concept of discretization and operates a uni-dimensional controlled random walk (RW) in the discretized space, to locate the unknown parameter. However, by moving to nonneighbor points in the space, our newly proposed hierarchical stochastic searching on the line (HSSL) solution performs such a controlled RW on the discretized space structured on a superimposed binary tree. We demonstrate that the HSSL solution is orders of magnitude faster than the original SPL solution proposed by Oommen. By a rigorous analysis, the HSSL is shown to be optimal if the effectiveness (or credibility) of the environment, given by *p*, is greater than the golden ratio conjugate. The solution has been both analytically solved and simulated, and the results obtained are extremely fascinating, as this is the first reported use of time reversibility in the analysis of stochastic learning. The learning automata extensions of the scheme are currently being investigated.**

***Index Terms*—Controlled random walk, discretized learning, learning automata, stochastic-point problem, time reversibility.**

Manuscript received April 17, 2013; revised October 17, 2013; accepted December 28, 2013. Date of publication February 17, 2014; date of current version October 13, 2014. The work of B. J. Oommen was supported by the Natural Sciences and Engineering Research Council of Canada. A preliminary version of this paper was presented at IEAAIE'12, the 2012 International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Dalian, China, in June 2012. This paper was recommended by Associate Editor S. Hu.

A. Yazidi is with the Department of Computer Science, Oslo and Akershus University College, Oslo 0167, Norway and earlier he was with Teknova AS, Sorlandet Kunnskapspark, Kristiansand 4630, Norway (e-mail: Anis.Yazidi@hioa.no).

O.-C. Granmo and M. Goodwin are with the Department of ICT, University of Agder, Grimstad 4630, Norway (e-mail: ole.granmo@uia.no; morten.goodwin@uia.no).

B. J. Oommen is with the School of Computer Science, Carleton University, Ottawa, ON K1S 5B6, Canada and also with the University of Agder, Grimstad 4630, Norway (e-mail: oommen@scs.carleton.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2014.2303712

[1]As we shall see later, hierarchical solutions have been proposed in the field of LA.

I. Introduction

In its most general form, any optimization problem can be characterized as being one which searches for a parameter (or a point in the appropriate space) which minimizes or maximizes a criterion function. If one considers the parameter that is sought for, as a point on the line, for example, we can model this problem as the so called stochastic point location (SPL) problem. This paper deals with a general solution to the SPL problem. However, as opposed to the existing solutions to the SPL problem, we resolve it by proposing a merging of two completely disjoint fields in computer science, those of directional stochastic optimization[2] and data structures.

1) The SPL problem: We can formally elucidate the SPL as follows. Consider the problem of a robot [algorithm, learning mechanism (LM)] moving along the real line attempting to locate a particular point λ^*. To assist the mechanism, we assume that it can communicate with an environment (Oracle), which guides it with information regarding the direction in which it should go. If the environment is deterministic, the problem is the deterministic point location problem, which has been studied rather thoroughly. In its pioneering version, Baeza-Yates *et al.* [6] presented the problem in a setting such that the environment could charge the robot a cost that is proportional to the distance it is from the point sought for. The question of having multiple communicating robots locate a point on the line has also been studied by Baeza-Yates *et al.* [6], [7]. Of course, vector-based versions of this problem lead to solutions

[2]Although hierarchical strategies have been earlier used in the field of learning automata (LA) [3]–[5], [12], [28], [30], [45], [55], they have not been used to resolve the SPL problem.

of the corresponding multidimensional optimization problems.

2) Optimization and the SPL problem:[3] While it is true that we and a few others [16] have referred to this problem as the SPL problem, the fact is that, essentially, every optimization problem deals with locating the optimal point within the domain of interest. Age-old techniques resort to moving toward the optimal point by using the first derivative of the criterion function, and where these updates are, for example, either proportional to the derivative(s) or involve the second derivative(s). In a related manner, the SPL is also associated to the field of learning automata (LA) in which the action space has an infinite number of actions [52]. In this case, one would not resort to invoking the derivatives of the criterion, but rather to sampling the action space and moving appropriately within the space. It is, however, one of the authors of this paper (Oommen) who coined SPL to distinguish it from the traditional class of optimization problems which explicitly looks at the derivatives. Rather, he considered the case where the directional information provided by an alternate source, for example, as inferred from the derivative, could suffice to achieve the optimization—without actually explicitly computing the first and higher-order derivatives. This is what distinguishes the SPL from the traditional class of optimization and LA problems. We should, however, emphasize that all the well known traditional methods could, just as well, be used to solve the SPL problem if the derivatives/gradients are available or if they can be inferred/approximated. Rather, we have chosen to not resort to the latter schemes because applying them in modern-day research does not involve either novelty or scientific risk.

   It is extremely pertinent and interesting to mention that the SPL, almost exactly as it is described by us and in [19], has been precisely the model used in selecting doses in clinical practice and experiments. The problem has been referred to as "The Design of Up and Down Clinical Trials" and is cited in [19]. The model is essentially the same as ours except that the solution scheme allows the algorithm to also stay at the same state at the next time instant. One should also observe that the proof and the concepts are almost the same as in the first proposed solution to the SPL [36]. It should also be mentioned that Kpamegan and Flournoy [19] suggest that the problem can be solved, as alluded to earlier, using stochastic optimization methods.

3) Relationship between the SPL and LA: The field of LA [13], [20], [29], [31], [34], [49], [56] deals with a learning machine attempting to learn the optimal action from a finite set of actions. LA, indeed, constitute the foundational basis for the field of reinforcement learning. It is pertinent to emphasize that the SPL problem generalizes the LA learning paradigm for the case when the optimal action can be an element from an infinite set.[4] More specifically, unlike the traditional LA model in which the LA attempts to learn the optimal action offered by the environment, we consider the following learning problem: the LM is trying to locate an unknown point on a real interval (and not just an optimal action from a set of actions) by interacting with the stochastic environment through a series of informed guesses. Thus the SPL problem, like the fundamental LA problem, is of importance in its own right, and also on the merit of the potential that it has in all LA-based applications.

4) The SPL and meta-learning: Before we proceed, it is pertinent to mention that the SPL can be considered to be a meta-learning algorithm in and of itself. Optimization algorithms, such as those alluded to above, typically, have a key meta-parameter that determines the convergence of the algorithm to the optimum. The choice of the value for this meta-parameter is therefore critical to the algorithm. In many cases, the meta-parameter of the scheme is related to the second derivative of the criterion function, which results in a technique analogous to a Newton's root solving scheme. The disadvantages of the latter are well known; if the starting point of the algorithm is not well chosen, the scheme can diverge; in addition, if the second derivative is small, the scheme is ill defined. Finally, such a scheme requires the additional computation involved in evaluating the (matrix of) second derivatives [47], [50], [59]. The application of the SPL to learn such a meta-parameter is straightforward.

5) Applications of the SPL: LA have been utilized in many real-life applications including power management in smart grids [27], distributed channel selection [60], solving the minimum weight connected dominating set [58], multiclass classification [1], power control [62], service selection [61], solving a large class of wireless networks related problems [33], a general class of stochastic decentralized games [57], adaptive control of antennas in wireless push networks [32], and in optimal sensor placement [8]. Without belaboring the point, the reader will see that in the light of the above, the SPL has applications in all these areas where LA have been utilized.

Apart from the LA-based applications, in the optimization scenario alluded to above, although the optimal point is reckoned to be unknown, one usually assumes the existence of an indicator as to the (approximate) value of the criterion function for any specified value of the parameter. Thus, the SPL has potential in all such optimization scenarios. Applications of such optimization methods (which are, really, all-pervasive) include the domains of image processing [46], pattern recognition [11], neural computing [14], economics [48], robotics [53], optimal control [10] and trajectory planning [22].

Apart from the above generic applications, the SPL has actually been specifically utilized in real-life problems that have to do with the processing of text in word-of-mouth

[3]We are thankful to two anonymous referees who requested this clarification—to primarily distinguish between these two families of problems.

[4]We should observe, though, that LA interacting with environments offering an infinite set of actions have also been studied [52]. However, the model of the response given by the environment in [52] is distinct from the model used here.

services, and in multidimensional scaling [37]. Indeed, without making much ado, we briefly state that if one had not invoked an SPL-based solution, the latter would have been a computationally infeasible problem. One should also observe that the problem of tracking changes in a time-varying system, which we have investigated here using the SPL solution, is also closely related to estimation in nonstationary environments. This problem has also been studied for decades, and even more recently for web-based applications [15].

Finally, as stated above, the SPL in its virgin form, has been precisely the model used in selecting doses in clinical practice and experiments to resolve the problem of designing up and down clinical trials [19].

II. Prior Art and Existing SPL Solutions

To place our work in the right perspective, we start this section by providing a brief review of the main concepts of the SPL problem as first introduced in [36]. We assume that there is a LM whose task is to determine the optimal value of some variable (or parameter), λ. We assume that there is an optimal choice for λ—an unknown value, say $\lambda^* \in [0, 1)$. The question which we study here is that of learning λ^*. Although the mechanism does not know the value of λ^*, we assume that it has responses from an intelligent environment, Ξ, which is capable of informing it whether any value of λ is too small or too big. To render the problem both meaningful and distinct from its deterministic version, we would like to emphasize that the response from this environment is assumed faulty. Thus, Ξ may tell us to increase λ when it should be decreased, and vice versa. However, to render the problem tangible, in [36] the probability of receiving an intelligent response was assumed to be $p > 0.5$, in which case Ξ was said to be informative. Note that the quantity p reflects on the effectiveness of the environment. Thus, whenever the current $\lambda < \lambda^*$, the environment correctly suggests that we increase λ with probability p. It simultaneously could have incorrectly recommended that we decrease λ with probability $(1-p)$. The converse is true for $\lambda \geq \lambda^*$.

The precursor to the SPL was the work of Bentley and Yao [9], who worked with the deterministic version of point location (i.e., where the environment provided fault-free directional responses) by superimposing the search problem onto a binary tree. The reader must observe that the SPL is far more complex than its deterministic version, because any algorithm which works with the latter could be led completely astray if the responses from the Oracle are stochastic.

A. State-of-the-Art

We can summarize the existing SPL-related literature as follows.

1) Oommen [36] pioneered the study of the SPL when he proposed and analyzed an algorithm that operates on a discretized search space[5] while interacting with an informative environment (i.e., $p > 0.5$). The search space is first sliced into N sub-intervals at the positions $\{0, \frac{1}{N}, \frac{2}{N}, \ldots, \frac{N-1}{N}, 1\}$, where a larger value of N will ultimately imply a more accurate convergence to λ^*. The algorithm then did a controlled random walk on this space by obediently following the environment's advice in the discretized space. In spite of the Oracle's erroneous feedback, this discretized solution was proven to be ϵ-optimal.
2) In the field of LA, Santharam *et al.* [52] presented an alternative nondiscretized solution to find the optimal action from an infinite number of actions in a continuous space. But the results presented in [52] are not directly applicable to the SPL.
3) An novel alternate parallel strategy[6] that combined LA and pruning was used in [41] to solve the SPL. By utilizing the response from Ξ, Oommen and Raghunath [41] partitioned the interval of search into three disjoint subintervals, eliminating at least one of the subintervals from further search, and by recursively searching the remaining interval(s) until the search interval was at least as small as the required resolution. This paper was subsequently enhanced in [42], where the authors introduced the Continuous Point Location with Adaptive d-ARY Search (CPL-AdS), which was another more efficient ϵ-optimal scheme. Huang and Jiang [16] proposed a rather straightforward modification of the latter CPL-AdS to also track changes in λ^*. Indeed, to achieve the latter, Huang and Jiang [16] proposed to perform an additional parallel d-ARY search at each epoch on the original search interval.[7] However, more importantly, the interesting facet of the solution presented in [42] is that it converges with an arbitrarily high accuracy even if the Oracle is a stochastic compulsive liar who is attempting to stochastically deceive the LM.
4) Oommen *et al.* [39] reported the first known solution to the SPL for a new model of nonstationarity referred to as metalevel nonstationarity. The question of analyzing our present HSSL scheme for the latter scenario is an open problem.
5) The use of a hierarchical mechanism to traverse the action space was earlier described in [3]–[5], [12], [28], [30], [45], [55]. Our hierarchical strategy is distinct from all of these, as is the modus operandus of the analysis, i.e., the time reversibility of the Markov chain.

III. Time Reversible Markov Chains

A fundamental contribution of this paper is the analysis of stochastic learning systems using the concepts of time reversibility. Since this is crucial to this paper, this phenomenon is briefly surveyed here. However, in order to fully comprehend this contribution, it is necessary for the reader

[5]Some of the existing results about discretized automata are found in [2], [21], [35], [38], [40], [43], [54]. Indeed, the fastest reported LAs are the discretized pursuit and maximum likelihood and Bayesian estimator algorithms [2], [40], [43].

[6]Put in a nutshell, in this paper, we aim to design a novel sequential (as opposed to parallel) hierarchical SPL solution.

[7]It was shown in [16] that the strategy can only track λ^* under certain conditions relative to the frequency of change in λ^* and the length of an epoch.

to understand the mathematical tools used in the prior art, as opposed to those which we propose.

A. Prior and New Mathematical Tools Used

1) Mathematical tools used in the prior art: The informed reader will observe that all the existing algorithms in stochastic learning and LA work with the concept of small-step processes. The reason for this is the most difficult part in the design and analysis of LA consists of the formal proofs of their convergence accuracies. The mathematical techniques used for the various families [fixed structure (FSSA), variable structure (VSSA), discretized, etc.] are quite distinct. The proof methodology for the family of FSSA is the simplest. It quite simply involves formulating the Markov chain for the LA, computing its equilibrium (or steady state) probabilities, and then computing the asymptotic action selection probabilities. In almost all these cases (except for the rare exception of the Krinsky LA), the state transitions are to neighboring states only. The proofs of convergence for VSSA are more complex and involve the theory of distance diminishing operators, and the theory of regular functions, all of which are effective by virtue of the small-step Markov processes involved. In other words, the LA must move from a point in the probability space to a point in its neighborhood for the convergence to be true. The proofs for discretized LA involve the asymptotic analysis of the Markov chain that represents the LA in the discretized space, whence the total probability of convergence to the various actions is evaluated. But in this case too, the discretized LA would move to a neighbor state so that the analysis of the Markov chain is feasible. This condition of moving to points close enough to the current probability vector is also true for the convergence of the family of the so called estimator algorithms.
2) Novel mathematical tools used: In this paper, we are attempting a completely new learning philosophy. Whenever we move within the probability space, we will choose to move to a point that could be significantly distant from the current point. This could, of course, imply huge perturbations of the present solution. But the amazing aspect of this is that since these large perturbations are done in a controlled manner, one can still obtain asymptotically-optimal convergence. The really fascinating issue here is that the convergence is, indeed, an order of magnitude faster than what one would obtain using a small-step paradigm. The reason for this enhanced speed is probably because of the fact that by making huge perturbations, the LM is able to quickly explore unexplored areas of the probability space, and discard the less crucial areas of the space without wasting too much time on them, thus resolving the exploitation versus exploration paradox.

From this perspective, this paper attempts to extend the horizon of methods that can be used to analyze LA. Indeed, this is the first reported use of time reversibility in the analysis of stochastic learning. The LA extensions of the scheme are currently being investigated, and we believe, hold vast potential.

The question of interest is really one of knowing how these huge perturbations are to be made. If these are done in an uncontrolled manner, it could lead to nothing more than a Monte-Carlo-type random sampling algorithm. But we advocate making these large perturbations based on an intelligent partitioning of the probability space by mapping it onto a binary tree. We motivate this by stating that we shall devise a counterpart tree-based search technique to solve the SPL problem. The rationale of the solution is to take advantage of the tree structure of the search space in order to enhance the search speed. This would enable the LM to quickly explore the search space and hopefully, focus its visits on the region that contains λ^*. More philosophically, we resolve this within the SPL framework by proposing a merging of two completely disjoint fields in computer science—those of stochastic optimization and data structures.

B. Time Reversibility

Certain specific Markov chains have the property that the process behaves in just the same way regardless of whether time is measured forward or backward. Kelly [18] made an analogy saying that if we take a film of such a process and then run the film backward, the resulting process will be statistically indistinguishable from the original process. This property is described formally in the following definition.

Definition 1: A stochastic process $X(t)$ is time reversible if a sequence of states $(X(t_1), X(t_2), \ldots, X(t_n))$ has the same distribution as the reversed sequence $(X(t_n), X(t_{n-1}), \ldots, X(t_1))$ for all $t_1, t_2, \ldots, t_n$. □

Consider a stationary Ergodic Markov chain (that is, a Markov chain that has been in operation for a long time) having transition probabilities M_{st} and stationary probabilities $P\{\pi_s\}$. Suppose that starting at some time we trace the sequence of states going backward in time. That is, starting at time t, consider the sequence of states $X_t, X_{t-1}, X_{t-2}, \ldots X_0$. It turns out that this sequence of states is itself a Markov chain with transition probabilities $Q_{st} = (P\{\pi_t\}/P\{\pi_s\}) * M_{ts}$. If $Q_{st} = M_{st}$ for all s, t, then the Markov chain is said to be time reversible. Note that the condition for time reversibility, namely, $Q_{st} = M_{st}$, can also be expressed as

$$P\{\pi_s\} M_{st} = P\{\pi_t\} M_{ts} \qquad \text{for all} \quad s \neq t. \tag{1}$$

The condition in the above equation can be stated as follows. For all states s and t, the rate at which the process goes from s to t (namely $P\{\pi_s\} M_{st}$) is equal to the rate at which the process goes from t to s (namely, $P\{\pi_t\} M_{ts}$). It is worth noting that this is an obvious necessary condition for time reversibility since a transition from s to t going backward in time is equivalent to a transition from t to s going forward in time. Thus, if $\pi_m = s$ and $\pi_{m-1} = t$, then a transition from s to t is observed if we are looking backward in time, and one from t to s is observed if we are looking forward in time.

The following theorem adapted from Ross and used universally [17], [18], [51] gives the necessary/sufficient condition for a finite ergodic Markov chain to be time reversible. Its proof is found in [51, p. 143].

Theorem 1: A finite ergodic Markov chain for which $M_{st} = 0$ whenever $M_{ts} = 0$ is time reversible if and only if starting in state s, any path back to s has the same probability as the reversed path. That is, if

$$M_{s,s_1} M_{s_1,s_2} \dots M_{s_k,s} = M_{s,s_k} M_{s_k,s_{k-1}} \dots M_{s_1,s}$$

for all states $s, s_1, \dots, s_k$.

□

Using the above theorem, we state the result that any tree structure associated with a finite stationary Markov process is time reversible. This follows from the avenue that a Markov chain resulting from the transition operations on any tree structure is time reversible. In fact, this result is not totally new. Kelly [18, p. 9] proved the following lemma. Although the lemma and its proof are fairly deep, one can get an intuitive sense for why it is true by observing that the shortest path between the nodes of a tree (say from A to B and from B to A) pass through a common ancestor, rendering the process of moving from A to B and from B to A time reversible.

Lemma 1: (Adapted from Kelly [18].) If the graph G associated with a stationary Markov process is a tree, then the process is time reversible. □

Although Kelly reported this result, he did not demonstrate how to associate a tree with a stationary Markov chain. In this paper, we shall give a formal definition for one such tree structure by organizing the points on the line along a tree, and prove the corresponding theorem regarding its time reversibility. The application of time reversibility in the domain of self-organizing lists has been reported elsewhere [44].

IV. Solution: Merging Field of Binary Search and SPL

In our proposed solution, the space of the search is arranged in the form of a binary tree with depth $D = log_2(N)$, where N is the resolution of the algorithm. The LM searches for the optimal value λ^* by orchestrating a controlled RW on a tree.

A. Definitions

Construction of Hierarchy. Let $\Delta = [\sigma, \gamma)$ be the current search interval containing λ^* whose left and right (smaller and greater) boundaries on the real line are σ and γ, respectively. Without loss of generality we assume that $\sigma = 0$ and $\gamma = 1$. The search space is constructed as follows: First of all, the hierarchy is organized as a complete binary tree with maximal depth D. To each node in the hierarchy we associate an interval. For convenience, we will use the same notation as in [12] and index the nodes using both their depth in the tree and their relative order with respect to the nodes located at the same the depth.

Root Node. The hierarchy root (at depth 0), which we call $S_{\{0,1\}}$, is assigned the interval $\Delta = \Delta_{\{0,1\}} = [0, 1)$. This interval is partitioned into two disjoint equi-sized[8] intervals $\Delta_{\{1,1\}}$ and $\Delta_{\{1,2\}}$, such that $\Delta_{1,1} = [0, 1/2)$ and $\Delta_{1,2} = [1/2, 1)$. Note that $1/2 = mid(\Delta_{\{0,1\}})$, where $mid(\Delta_{\{0,1\}})$ denotes the midpoint of $\Delta_{\{0,1\}}$. We shall simultaneously use the notation[9] and refer to the interval $\Delta_{\{1,1\}}$ as the Left Child of the root and to $\Delta_{\{1,2\}}$ as its Right Child.

Nodes at Depth d. Node $j \in \{1, ..., 2^d\}$ at depth d, called $S_{\{d,j\}}$, where $0 < d < D$, is assigned the interval $\Delta_{\{d,j\}} = [\sigma_{\{d,j\}}, \gamma_{\{d,j\}})$ which is partitioned into two disjoint equi-sized intervals $\Delta_{\{d+1,2j-1\}}$ and $\Delta_{\{d+1,2j\}}$. Following the same previously alluded to nomenclature, $\Delta_{\{d+1,2j-1\}}$ is the Left Child of $\Delta_{\{d,j\}}$ and $\Delta_{\{d+1,2j\}}$ is its Right Child.

Nodes at Depth D. At depth D, which represents the maximal depth of the tree, the nodes do not have children. In fact, when the search interval is at least as small as the required resolution of estimation, we cannot perform additional partitioning. Observe that by virtue of the equi-partitioning property, for a given node j at depth d attached to the respective interval $\Delta_{\{d,j\}}$, we can deduce the values of the left and right boundaries of the interval: $\sigma_{\{d,j\}} = (j-1)(\frac{1}{2})^d$ and $\gamma_{d,j} = j(\frac{1}{2})^d$, for $j \in \{1, ..., 2^d\}$ where $0 \le d \le D$.

Convention Regarding the Root's Notation. Since a level of value "-1" is nonexistent, we use a boundary notation and denote the Parent of $\Delta_{\{0,1\}}$ to be $\Delta_{\{0,1\}}$ itself. The same applies to the root node. In other words, Parent($\Delta_{\{0,1\}}$)=$\Delta_{\{0,1\}}$.

Convention Regarding the Leaves' Notation. In a same vein, since level $D + 1$ is nonexistent, we use the convention that Right Child of a leaf node is the same as the leaf node in question itself. Similarly, the Left Child of a leaf node is the leaf node itself. Formally, we say that

$$Left\ Child\,(S_{\{D,j\}}) = Right\ Child(S_{\{D,j\}}) = S_{\{D,j\}} \quad \text{for } j \in \{1, ..., 2^D\}.$$

Target Node. We define the target node as the leaf node whose associated interval contains λ^*.

Nontarget Node. These are the leaf nodes whose corresponding assigned intervals do not contain λ^*.

Resolution. We refer to the scheme's resolution to denote the number of leaf nodes that the scheme has, i.e, $N = 2^D$. Whenever the learner is at a certain node in the tree, we propose to use the mid-point of the interval itself as an estimate of the unknown λ^*. By virtue of the equi-partitioning of the intervals at each level of the tree, whenever the LM is at a node of a certain depth d in the tree, the estimate of λ^* will take a discretized value that is a multiple of $(\frac{1}{2})^{d+1}$. More precisely, whenever the LM is at a leaf node, the estimate of λ^* will take a discretized value among the following N discretized values:

$$\{mid(\Delta_{\{D,1\}}), mid(\Delta_{\{D,2\}}), \dots, mid(\Delta_{\{D,N\}}\} = \left\{ \left(\frac{1}{2}\right)^{D+1}, 3\left(\frac{1}{2}\right)^{D+1}, 5\left(\frac{1}{2}\right)^{D+1}, \dots, (2N-1)\left(\frac{1}{2}\right)^{D+1} \right\}.$$

[8]The equi-partitioning is really not a restriction. This can be easily generalized.

[9]For the rest of the paper, to prevent confusion, since the intervals and their values are interchangeable, we refer to Parent, Left Child, and Right Child of an interval $\Delta_{\{i,j\}}$ as the interval associated to the respective Parent, Left Child, and Right Child of the node $S_{\{i,j\}}$, respectively.

Using some simple algebraic manipulations, these discretized values can be expressed as

$$\left\{\frac{1}{N}-\delta_N, \frac{2}{N}-\delta_N, \frac{3}{N}-\delta_N, \dots, \frac{N-1}{N}-\delta_N, \frac{N}{N}-\delta_N\right\}$$

where

$$\delta_N = \frac{1}{2N}.$$

The reader must note that, at the leaf nodes, the possible values that the estimate of λ^* can take are equi-spaced with a fixed step size $\frac{1}{N}$. He will thus see that such a discretization is distinct from the discretization used in discretized LA and in the original SPL solution [36] since the values 0 and 1 are not included.

The same explanation applies at intermediate nodes at any level d; in such cases our estimate of λ^* is also a discretized value but from a different set of $N_d = 2^d$ discretized possible values. Thus, whenever LM is at a certain node of depth d where $0 \le d \le D$, the estimate of λ^* will take a discretized value from among the N_d following discretized values:

$$\begin{aligned}&\left\{mid(\Delta_{\{d,1\}}), mid(\Delta_{\{d,2\}}), \dots, mid(\Delta_{\{d,N_d\}})\right\}\\ &= \left\{\frac{1}{N_d}-\delta_{N_d}, \frac{2}{N_d}-\delta_{N_d}, \frac{3}{N_d}-\delta_{N_d}, \dots, \frac{N_d-1}{N_d}-\delta_{N_d}, \frac{N_d}{N_d}-\delta_{N_d}\right\}\end{aligned}$$

where

$$\delta_{N_d} = \frac{1}{2N_d}.$$

The reader should note that the term resolution itself is relative. It becomes finer at deeper levels of the tree's hierarchy.

B. Structure of Search Space and Responses from Ξ

We intend to organize the search space in the form of a complete binary tree, where each node corresponds to an interval range. Initially, and at every step, we guess the midpoint of the given interval to be our estimate of the unknown λ^*. The LM searches for the optimal value λ^* by operating a random walk on the tree, moving from one tree node to another.

As shown in Fig. 1, where the unit interval is partitioned into eight subintervals, each node in the tree is associated with an interval; e.g., the root is associated with the interval $[0, 1)$. This interval is partitioned into two disjoint equi-sized intervals. In this setting, the left child of the root is associated with $[0, 1/2)$, the right child with $[1/2, 1)$, and so on. As alluded to previously, we use as an estimate of the unknown λ^*, the middle point of the interval associated with the node where the LM resides.

At any given time instance, the LM finds itself at a node $S_{\{d,j\}}$ in the tree, where $j \in \{1, \dots, 2^d\}$ and $0 \le d \le D$. It then attempts to infer the next promising search interval that is likely to contain λ^* by making a sequence of informed guesses. Observe though that for each guess, the environment Ξ (Oracle) essentially informs the LM, possibly erroneously (i.e., with probability p), which way it should move to reach the unknown point. Let $\Delta_{\{d,j\}}$ be the interval that is associated with the node where the LM resides at the current time instant. The informed guesses correspond to a sampling at the boundary points of the interval $\Delta_{\{d,j\}}$, and at the midpoint of the interval: $mid(\Delta_{\{d,j\}})$.

In this sense, the set of sampled points is expressed as a vector $\overrightarrow{x} = [x^1, x^2, x^3]$, where

$$x^1 = \sigma_{\{d,j\}} = (j-1)\left(\frac{1}{2}\right)^d, \; x^2 = mid(\Delta_{\{d,j\}}) = (2j-1)\left(\frac{1}{2}\right)^{d+1}$$

$$\text{and } x^3 = \gamma_{\{d,j\}} = j\left(\frac{1}{2}\right)^d.$$

The corresponding response of the environment Ξ can be formulated as a tuple $\overrightarrow{\Omega} = [\Omega^1, \Omega^2, \Omega^3]$.

Ω^k, for $k \in \{1, 2, 3\}$, is a random variable that can take either the value Left or Right. We will use L to imply a region to the Left of the respective sampled point, and R to imply a point to the Right of the respective sampled point. Since the environment Ξ is assumed faulty, we suppose that it suggests the correct direction with a probability p. Therefore, Ω^k, for $k \in \{1, 2, 3\}$ can be formally defined based to whether λ^* is larger than or smaller than x^k.

If $\lambda^* < x^k$

$$\Omega^k = \begin{cases} L & \text{with probability } p \\ R & \text{with probability } (1-p). \end{cases}$$

If $\lambda^* \ge x^k$

$$\Omega^k = \begin{cases} L & \text{with probability } (1-p) \\ R & \text{with probability } p. \end{cases}$$

Therefore, considering the three distinct sampling positions, the environment Ξ responds with one of the 2^3 possible results: {[L, L, L], [L, L, R], [L, R, L], [L, R, R], [R, L, L], [R, L, R], [R, R, L], [R, R, R]}.

Remark 1: As in [41], the environment's feedback $\overrightarrow{\Omega} = [\Omega^1, \Omega^2, \Omega^3]$ will be reckoned to be inconsistent if

1) $\Omega^i = Left$ and $\Omega^j = Right$ where $i < j$. In other words, the environment contradicts itself by suggesting that λ^* is less than x^i while being bigger than x^j. Since $x^i < x^j$ for $i < j$, by virtue of the construction of the partitioning, the feedback should, indeed, be deemed to be inconsistent since there is no real number λ^* that simultaneously satisfies the pair of contradictory inequalities, namely, $\lambda^* \le x^i < x^j$ and at the same time $x^i < x^j \le \lambda^*$.

Therefore, [L, R, R], [L, L, R], [R, L, R], and [L, R, L] are considered inconsistent.

C. Mapping Responses to Transitions

The estimated value for λ^* is the midpoint of the interval associated with the current node in which the LM resides. The crucial issue that we address in this section is that of determining how to change our current guess λ of the unknown λ^* based on the faulty nature of the response from the Oracle. From this perspective, we seek a procedure that decides the next LMs search interval, which ultimately boils downs to designing a set of rules that control the LMs moves in such a way that it advances toward the next promising node in the tree (i.e., the one associated with an interval that is likely to contain the unknown λ^*).

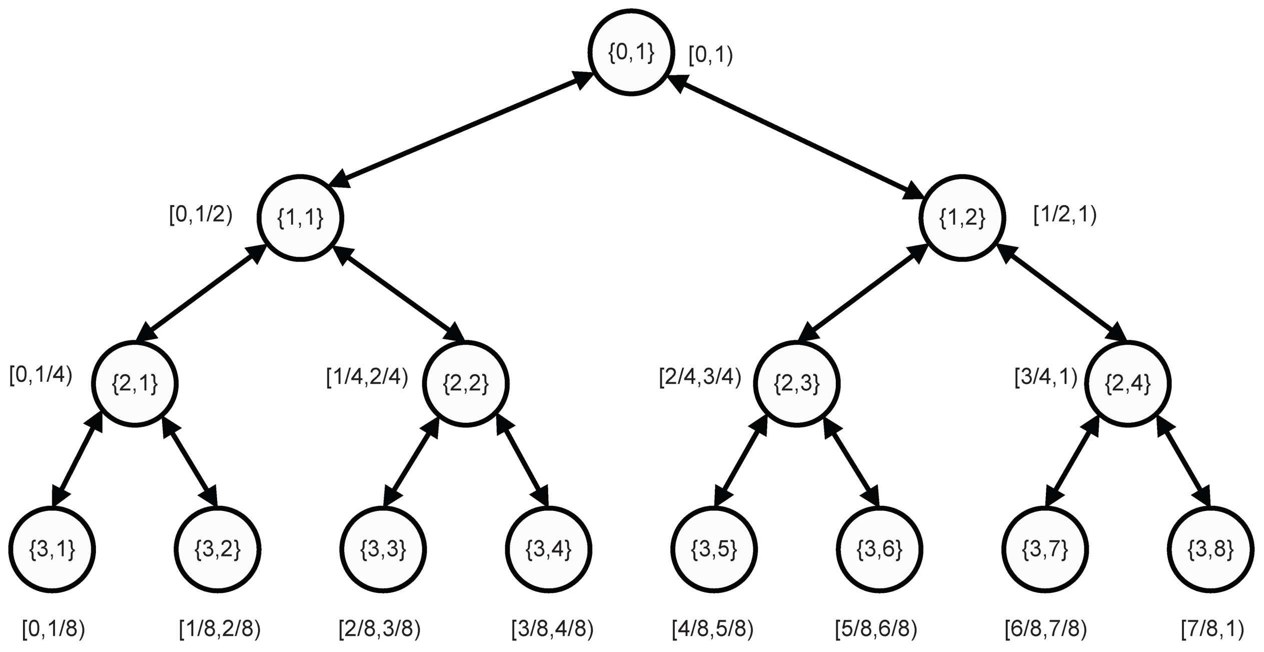


Fig. 1. Search space is organized as a tree. Each node $S_{\{i,j\}}$ in the tree is associated with an interval $\Delta_{\{i,j\}} = [\sigma_{\{i,j\}}, \gamma_{\{i,j\}})$. This interval is sampled at $\sigma_{\{i,j\}}$, $\gamma_{\{i,j\}}$, and $mid(\Delta_{\{i,j\}})$, producing one of the eight possible responses: {[L, L, L], [L, L, R], [L, R, L], [L, R, R], [R, L, L], [R, L, R], [R, R, L], [R, R, R]}.

TABLE I
DECISION TABLE TO CHOOSE THE NEXT SEARCH INTERVAL BASED ON THE RESPONSE VECTOR $[\Omega^1, \Omega^2, \Omega^3]$, WHEN THE CURRENT SEARCH INTERVAL IS $\Delta_{\{i,j\}}$

Next Search Interval	Condition
$Parent(\Delta_{\{i,j\}})$	[R, R, R] $\bigvee$ [L, R, R] $\bigvee$ [L, L, R] $\bigvee$ [L, L, L]
$LeftChild(\Delta_{\{i,j\}})$	[R, L, R] $\bigvee$ [R, L, L]
$RightChild(\Delta_{\{i,j\}})$	[R, R, L] $\bigvee$ [L, R, L]

Based on these responses, we propose that the LM move to another (nonneighbor) node, either to the current node's parent, or to one of its children (Left Child/Right Child). The rules for moving in the tree are summarized in Table I.

The reader should observe that we are explicitly making use of the Oracle's feedback in Table I—even if it is inconsistent. The reason why we have opted to include the inconsistent feedback in the table is to remain consistent with the original SPL algorithm by allowing movements at each time instant in the search space and suppressing self loops in intermediate nodes at the next time instant. In fact, in the original SPL solution, the LM always makes a step (forward or backward) and never stays at an intermediate node. The same design principles were applied for the HSSL where the LM moves to explore another node in the tree even when the feedback is inconsistent.[10]

[10]At this juncture, we should mention that we foresee a strong analogy between deciphering the faulty responses of the Oracle and the theory of error-correcting codes [26], where, in the latter field, the possible transmitted message is inferred from its noisy version by analyzing/comparing it with a set of codewords [26]. A transmission error is detected if the received message does not match any codeword. Such a study, which incorporates error-correcting Codes into the HSSL, is beyond the scope of this paper. It is currently being investigated.

We also mention, in passing, that preliminary analytical and empirical results are available for the case when the LM only makes use of the Oracle's consistent feedback and ignores inconsistent feedback. However, in the interest of brevity, these results are not included in this paper.

Notice that although the above transition rules catalogued in Table I are deterministic, because the environment is assumed faulty, the state transitions of the underlying Markov chain are stochastic. Further, observe that we propose two types of random walk transitions in the tree.

1) *Reverse transitions*: Transitions of this type correspond to a movement to a lower level in the hierarchy. This happens when the LM moves to the immediate Parent, implying a larger search interval, which, in turn, allows the LM to escape from getting trapped in a wrong subtree, i.e., one not containing λ^*.
2) *Top-down transitions*: Transitions of this type correspond to a movement to a deeper level in the hierarchy. Whenever the LM performs a transition to a deeper level in the hierarchy by choosing a Child node, the search space shrinks, and will, hopefully, concentrate on one of the contiguous intervals at the next level of the tree that contains λ^*.

The overall scheme is given in Algorithm 1 titled Algorithm HSSL.

Observe that the algorithm does not have an explicit terminating condition because being ergodic, it effectively Repeats ForEver. This is because, while the algorithm aims at converging to the optimal value of λ^* if the environment is stationary, it must also possess the ability to migrate from the point it has converged to and move toward a new value of λ^* if the environment is nonstationary. We also observe that when one examines the algorithm, one sees that we can

Algorithm 1 Algorithm HSSL

```
Input:   N = 2^D scheme resolution, p .
Output:  λ(n): time dependent estimate of λ*
Begin Algorithm HSSL
1: $\Delta = [\sigma, \gamma)$ /*Initial Search Interval*/
2: $CurrentSearchInterval = \Delta$
3: for Every time instant $n$ do
4:     $\lambda(n) = mid(CurrentSearchInterval)$
5:     Get $\sigma$, $\gamma$: the boundary points of $CurrentSearchInterval$
6:     $\overrightarrow{\Omega} = [\Omega^1, \Omega^2, \Omega^3]$ – Responses to the sampled points
       $[\sigma, mid(CurrentSearchInterval), \gamma]$.
7:     CurrentSearchInterval=NextSearchInterval(CurrentSearchInterval, $\overrightarrow{\Omega}$)
       /*Apply the Decision Table I to recursively choose the next search Interval*/
8: end for
End Algorithm HSSL
```

```
Procedure NextSearchInterval(SearchInterval, $\overrightarrow{\Omega}$)
Input:   SearchInterval, $\overrightarrow{\Omega}$: Corresponding vector of responses from the environment.
Output:  The new search interval to be processed.
Begin Procedure NextSearchInterval
1: Implement the Decision Table: Table I.
End Procedure NextSearchInterval
```

guarantee convergence to the desired resolution whenever we reach a leaf node. However, the scheme does converge at every non-leaf node too, albeit to a coarser resolution, namely to the resolution dictated by the subtree rooted at that particular node.

This concludes the description of our learning algorithm. We now investigate its convergence properties.

D. Analysis of Solution

In this section, we shall prove that the HSSL solution is asymptotically optimal. We shall show that, eventually, based on an informed series of guesses, the LM will be able to concentrate its moves within nodes in the tree that are associated with small intervals containing the optimal value λ^*, and that this will be true if p is larger than the conjugate of the golden ratio.[11]

With regard to the proof, we submit the following remarks regarding the transition probabilities and the condition of optimality.

1) The Golden Ratio: We denote by Φ the quantity described as the conjugate of the golden ratio, [24], where $\Phi = \frac{\sqrt{5}-1}{2} \approx 0.61803$. We emphasize that the fact that the optimality condition involves Φ is not really a limitation. To clarify this, suppose that the environment is informative, but its effectiveness p is less than Φ. The constraint that p is less than Φ can be countered by applying a majority voting algorithm. If p is known to the LM, this reduces to determining the minimum number of queries that one has to ask of the Oracle to ensure that the probability that the majority of responses being correct is larger than Φ. It should be obvious to the reader that a similar reasoning can be applied if p is unknown to the LM, and we only know a lower bound p_{min} of p such that $p_{min} > 0.5$. We will not elaborate on these ideas here.
It is also worth noting that the golden ratio and its conjugate are encountered in nature as well, as in many other mathematical and art-related problems [24].

2) The Transition Probabilities: Before we proceed, it is necessary for us to obtain the transition probabilities of the underlying Markov chain. We have to do this for every single transition, and this is, indeed, a rather laborious process. But to demonstrate how this is done, we shall show how they can be derived for a specific node in the tree. The transitions for the rest of the nodes will be merely written down and the underlying algebra will be omitted in the interest of avoiding repetition.

We denote Ω^* to be the correct response obtained from a nonfaulty environment (i.e., an environment for which $p = 1$). We now consider the case displayed in in the example illustrated in Fig. 2, where node $S_{\{3,7\}}$ is the target node. Thus, in this particular example, $\lambda^* \in [6/8, 7/8)$. Consider now the transitions at node $S_{\{1,2\}}$. Its associated interval, $[1/2, 1)$, is sampled at the points $\overrightarrow{x} = [1, 3/4, 1]$. Taking into account that $\lambda^* \in [6/8, 7/8)$, one can easily see that $\Omega^* = [R, R, L]$.

We now apply the rules for moving within the tree as summarized in Table I. Using these we will be able to derive the transition probabilities for moving to the parent node, right child node and left child nodes, respectively

$$\begin{aligned} p_{\{1,2\},\{0,1\}} &= Pr(\Omega = [R, R, R] \vee [L, R, R] \vee [L, L, R] \\ &\qquad \vee [L, L, L] | \Omega^* = [R, R, L]) \\ &= Pr(\Omega = [R, R, R] | \Omega^* = [R, R, L]) + Pr(\Omega \\ &= [L, R, R] | \Omega^* = [R, R, L]) \\ &\quad + Pr(\Omega = [L, L, R] | \Omega^* = [R, R, L]) + Pr(\Omega \\ &= [L, L, L] | \Omega^* = [R, R, L]) \\ &= p^2 q + q^2 p + q^3 + pq^2 = pq(p+q) \\ &\quad + q^2(p+q) = pq + q^2 \\ &= q(p+q) = q \end{aligned}$$

$$\begin{aligned} p_{\{1,2\},\{2,4\}} &= Pr(\Omega = [R, R, L] \vee [L, R, L] | \Omega^* = [R, R, L]) \\ &= Pr(\Omega = [R, R, L] | \Omega^* = [R, R, L]) \\ &\quad + Pr(\Omega = [L, R, L] | \Omega^* = [R, R, L]) \\ &= p^3 + qp^2 \\ &= p^2(p+q) \\ &= p^2 \end{aligned}$$

$$\begin{aligned} p_{\{1,2\},\{2,3\}} &= Pr(\Omega = [R, L, R] \vee [R, L, L] | \Omega^* = [R, R, L]) \\ &= Pr(\Omega = [R, L, R] | \Omega^* = [R, R, L]) \\ &\quad + Pr(\Omega = [R, L, L] | \Omega^* = [R, R, L]) \\ &= pq^2 + p^2 q \\ &= pq(p+q) \\ &= pq. \end{aligned}$$

[11]Throughout this section, we shall use that notation that $q = 1 - p$.

These are precisely the expressions that we will encounter later.

Theorem 2: The parameter learning algorithm specified by Algorithm 1 and the rules summarized in Table I are asymptotically optimal if p is larger than the conjugate of the golden ratio. Thus, we formally affirm that if $p > \Phi$: $Lim_{N\to\infty} Lim_{n\to\infty} E[\lambda(n)] \to \lambda^*$.

Proof: Our intention is to prove that as N is increased indefinitely (or equivalently D increased indefinitely), $Lim_{N\to\infty} Lim_{n\to\infty} E[\lambda(n)] \to \lambda^*$ whenever $p > \Phi$. We shall prove this by analyzing the properties of the underlying Markov chain which is specified by the rules in Table I.

Let H be the corresponding transition matrix formed by the rules in Table I. Clearly, H represents a single closed communicating class whose periodicity is unity. The chain is ergodic, and the limiting probability vector is given by the eigenvector of H^T corresponding to eigenvalue unity.

Let this vector be $\Pi = [\pi_{\{0,1\}}, \pi_{\{1,1\}}, \pi_{\{1,2\}}, \dots, \pi_{\{D,1\}}, \pi_{\{D,2\}}, \dots, \pi_{\{D,2^D\}}]$. Then, Π satisfies

$$H^T \Pi = \Pi. \tag{2}$$

Since the tree is a complete binary tree, in total, the tree consists of $2^{D+1} - 1$ nodes (since $2^{D+1} - 1 = 1 + 2 + ... + 2^D$).

We shall now specify the elements of the transition matrix, H. For each node in the tree, we shall derive the expression for the transition probabilities[12] to the next states (nodes). To achieve this task, we distinguish three cases, namely, whether the considered node is a root node, intermediate node or a leaf node.

a) *Transitions at the root node:* Consider the root node $S_{\{0,1\}}$. For the reflexive transition (from the root node to the root node itself), we have

$$p_{\{0,1\},\{0,1\}} = 1 - p^2 - pq = q.$$

Concerning the transitions to the children nodes, two cases emerge based on the relative positions of λ^* when compared to $mid(\Delta_{\{0,1\}})$.

1) If $0 \le \lambda^* < \frac{1}{2}$, then
$p_{\{0,1\},\{1,1\}} = p^2$
$p_{\{0,1\},\{1,2\}} = pq$.
2) If $\frac{1}{2} \le \lambda^* < 1$, then
$p_{\{0,1\},\{1,1\}} = pq$
$p_{\{0,1\},\{1,2\}} = p^2$.

b) *Transitions at intermediate nodes:* Consider an intermediate node $S_{\{d,j\}}$, i.e, node $j \in \{1, \dots, 2^d\}$ at depth d where $0 < d < D$. In order to specify the transitions probabilities at an intermediate node, we have to consider the following three cases.

1) If $\lambda^* \notin \Delta_{\{d,j\}}$, then
$p_{\{d,j\},\{d-1,\lceil j/2\rceil\}} = p^2 + q^2$
$p_{\{d,j\},\{d+1,2j-1\}} = pq$
$p_{\{d,j\},\{d+1,2j\}} = pq$.
2) If $\sigma_{\{d,j\}} \le \lambda^* < mid(\Delta_{\{d,j\}})$, then
$p_{\{d,j\},\{d-1,\lceil j/2\rceil\}} = pq + q^2 = q$
$p_{\{d,j\},\{d+1,2j-1\}} = p^2$
$p_{\{d,j\},\{d+1,2j\}} = pq$.
3) If $mid(\Delta_{\{d,j\}}) \le \lambda^* < \gamma_{\{d,j\}}$, then
$p_{\{d,j\},\{d-1,\lceil j/2\rceil\}} = pq + q^2 = q$
$p_{\{d,j\},\{d+1,2j-1\}} = pq$
$p_{\{d,j\},\{d+1,2j\}} = p^2$.

[12]As mentioned in the preface to the theorem, we shall not *derive* each of the transition probabilities. Rather we shall use the same arguments used there and merely write out the relevant expressions.

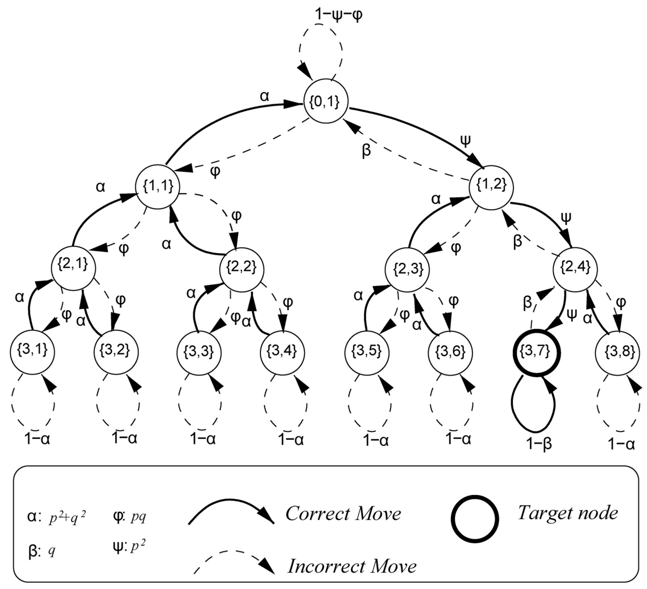


Fig. 2. Transition matrix H of the Markov chain formed by the rules in Table I, specified as a tree for the partitions described in Fig. 1.

c) *Transitions at the leaf nodes:* The transitions probabilities at a leaf node depend on whether it is a target node or a nontarget node. We consider each of these cases individually.

1) For a nontarget leaf node
$p_{\{D,j\},\{D,j\}} = 1 - (p^2 + q^2)$
$p_{\{D,j\},\{D-1,\lceil j/2\rceil\}} = p^2 + q^2$.
2) For the target node
$p_{\{D,i\},\{D,i\}} = p$
$p_{\{D,i\},\{D-1,\lceil i/2\rceil\}} = q$.

For the sake of clarity, we present a graphical example for the case when the depth of the tree is 3. Fig. 2 specifies the transition matrix of the associated Markov chain when node $S_{\{3,7\}}$ is the target leaf node.

The main idea of proof is to demonstrate that the limiting probability increases geometrically with the state indices along the shortest path to the target node. As a consequence of this, the limiting probability can be shown to be concentrated within an arbitrarily small interval around λ^*, simply by increasing the size of the tree.

We remind the reader that the target node, is the leaf node (at depth D), where the value λ^* is contained in its associated interval. Let i_D^* be the relative index of the target node among the leaf nodes which are, in turn, located at level D.

Let $Q_{\{0,1\},\{D,i_D^*\}}$ denote the shortest path in the balanced tree from the root node $S_{\{0,1\}}$ to the target node $S_{\{D,i_D^*\}}$.

$Q_{\{0,1\},\{D,i_D^*\}}$ is composed of the sequence of the indexes of the nodes leading to the target node $S_{\{D,i_D^*\}}$ when starting from the root $S_{\{0,1\}}$. Thus $Q_{\{0,1\},\{D,i_D^*\}} = [\{0,1\}, \{1,i_1^*\}, \dots, \{l,i_d^*\}, \dots, \{D,i_D^*\}]$ where i_d^* is the relative index of the node at depth d that belongs to the optimal path $Q_{\{0,1\},\{D,i_D^*\}}$. Equivalently, $S_{\{l,i_d^*\}}$ is the node at depth d ($0 \le d \le D$) that is on the path that starts from the root node and that leads to the target node. Clearly, $i_0^* = 1$.

Let $S_{\{D,j\}}$ be a nontarget leaf node. Also, let $Q_{\{D,j\},\{D,i_D^*\}}$ be the shortest path in the tree that connects the nontarget node $S_{\{D,j\}}$ to the target node $S_{\{D,i_D^*\}}$.

Following the tree structure of the search space, the path $Q_{\{D,j\},\{D,i_D^*\}}$ is clearly composed by the concatenation of the two following subpaths.

1) A subpath that originates from the nontarget node $S_{\{D,j\}}$ and that does not intersect with the path $Q_{\{0,1\},\{D,i_D^*\}}$.
2) A second subpath that intersects with $Q_{\{0,1\},\{D,i_D^*\}}$ (or more exactly a subpath of $Q_{\{0,1\},\{D,i_D^*\}}$). Let $S_{\{k,i_k^*\}}$ be the node whose index comes first in the ordered list of node indexes forming this subpath (i.e., the head of the sequence).

Therefore, $Q_{\{D,j\},\{D,i_D^*\}}$ can be seen as the concatenation of $Q_{\{D,j\},\{k,i_k^*\}}$ and $Q_{\{k,i_k^*\},\{D,i_D^*\}}$.

Informally speaking, moving along $Q_{\{D,j\},\{D,i_D^*\}}$ involves performing a series of reverse transitions from the nontarget leaf node to the first node whose associated interval contains λ^*, which has the effect of increasing the length of search space interval by a factor of two at each step, and then performing top-down transitions in the direction of the target leaf node, which, in turn, shrinks the length of the search interval by a factor two at each step, until the target node is attained.

Let us first study the transitions along the reverse path $Q_{\{D,j\},\{k,i_k^*\}}$. Examining the balance (equilibrium) equation of the Markov chain at the nontarget leaf node $S_{\{D,j\}}$ gives

$$\pi_{\{D,i\}} = (1 - p^2 - q^2)\pi_{\{D,i\}} + pq\pi_{\{D-1,\lceil i/2\rceil\}}.$$

We now observe that the Markov chain is time reversible [18]. We thus resort to the time reversibility property in order to deduce the following equation:

$$\pi_{\{d-1,\lceil i/2\rceil\}} = \frac{p^2+q^2}{pq}\pi_{\{d,i\}} \tag{3}$$

where d denotes any given level in the tree such that $k \le d \le D$.

Similarly, we consider the transitions along the path $Q_{\{k,i_k^*\},\{D,i_D^*\}}$. Examining the balance (equilibrium) equation of the Markov chain at the nontarget node $S_{\{D,i_D^*\}}$ gives

$$\pi_{\{D,i_D^*\}} = (1 - pq - q^2)\pi_{\{D,i_D^*\}} + p^2\pi_{\{D-1,\lceil i_D^*/2\rceil\}}.$$

Hence, we deduce that

$$\pi_{\{D,i_D^*\}} = \frac{p^2}{q}\pi_{\{D-1,\lceil i_D^*/2\rceil\}}.$$

By making use of the time reversibility property of the Markov chain, we can easily deduce that along the top-down path $Q_{\{k,i_k^*\},\{D,i_D^*\}}$, for $k \le d \le D$

$$\pi_{\{d,i_d^*\}} = \frac{p^2}{q}\pi_{\{d-1,\lceil i_d^*/2\rceil\}}. \tag{4}$$

We define z_1 as $z_1 = \frac{p^2+q^2}{pq}$. Clearly, the latter quantity, z_1, is always greater than 1 for any value of p since we know that $p^2 + q^2 > pq$. In the same manner, we denote by z_2 the quantity $\frac{p^2}{q}$. The inequality $z_2 > 1$ is equivalent to $p^2 > q$, which reduces to the constraint that p must be greater than the golden ratio.

With these balance (equilibrium) equations in place, we are ready to deduce the relationship that relates the stationary probability of the target node to the stationary probability of any nontarget node. Combining (3) and (4), and applying the reasoning behind the recurrence relationships we see that

$$\pi_{\{D,i_D^*\}} = z^{D-k}\pi_{\{D,j\}}$$

where $z = z_1 z_2$. Note that since we have $z_1, z_2 > 1$, then $z > 1$.

To finalize the proof, we consider the equilibrium (asymptotic) value of $E[\lambda(n)]$ for any finite depth D. To do this, we invoke arguments similar to those used in [36]. Since $z > 1$, the limiting probabilities Π increases geometrically with the state indices, along the shortest path to the target node, until its reaches its maximum at $\pi_{\{D,i_D^*\}}$. Since z^D increases exponentially and we are speaking about the mean of an increasing geometric progression, most of the mass will be concentrated among an arbitrarily small number of states close to the target node $S_{\{D,i_D^*\}}$.

Thus, as N goes to infinity the mean of $E[\lambda(\infty)]$ will be centered within a small interval $\Delta_{\{D,i_D^*\}}$ and will thus be arbitrarily close to λ^*. Hence the theorem. ■

Remark 2: One would have thought that as in the original solution to the SPL, the condition $p > 0.5$, that renders the environment to be informative, would have been sufficient for the convergence. But it appears as if this is the price that we have to pay. By making moves to non-neighbor points we can improve the convergence speed of the algorithm. But this comes with a small price, i.e., the condition for convergence becomes $p > 0.61803$ instead of $p > 0.5$. It is surely a conundrum as to why this golden ratio conjugate appears in the picture mysteriously, but, as we see, that is exactly how the mathematics ultimately works out. However, as explained in the opening paragraph of Section IV-D, this is really not a handicap.

V. Advantages Over State-of-Art Solutions

Based on the above, we can record the drawbacks of the prior art which we have remedied here.

1) One primary drawback of the state-of-the-art solutions to the SPL problem reported in [36], [41], and [42], and that we have remedied, is that the steps are always very conservative. If the step size is increased, the scheme converges faster, but the accuracy is correspondingly decreased, and vice versa. As the next section demonstrates, this paper has solved the SPL by proposing a solution that is an order of magnitude faster than the state-of-the-art, and this is done by incorporating completely different techniques, namely those similar to those used by Bentley and Yao [9] (which can be perceived as an expandable binary search) in solving deterministic point location. One can easily see that

generalizing the results of [9] for a stochastic setting is, indeed, far from trivial, which is why the reader will observe that our solution is totally distinct.

2) The first inherent shortcoming of the two algorithms proposed in [41] and [42] is that both work with the premise that λ^* is constant over time. In contrast, as we will show presently, our proposed HSSL is able to cope with nonstationary settings, i.e., where λ^* is time varying, without invoking an additional layer of CPL-AdS [16].
3) The second shortcoming of the schemes presented in [41] and [42] is that they required three (or d) LA operating, as it were, in parallel. We have not required such a parallelized mode of operation.
4) Another rather subtle issue that one encounters in the two algorithms presented in [41] and [42] is that they are error prone due to the fact that they rely on the ϵ-optimal property of the individual LA. In fact, in theory, at each epoch of the algorithms presented in [41] and [42], one is required to run each individual LA for an infinite number of iterations to ensure its convergence with probability 1 to its optimal action.[13] Therefore, the smaller the number of iterations, the higher is the probability by which it will converge to a wrong interval and thus discard the interval of interest that contains λ^*. Hence, in order to increase the confidence of the search procedure at each epoch, a considerable number of iterations per epoch is required, resulting in a diminished convergence speed. We have successfully eliminated this feature.
5) As mentioned earlier, our hierarchical strategy, the HSSL, is distinct from the one used in [3]–[5], [12], [28], [30], [45], and [55] for LA, as is the *modus operandus* of the analysis, i.e., the time reversibility of the Markov chain.

VI. Simulation Results

The LM described in this paper was experimentally evaluated to verify the validity of our analytic results and to examine its rate of convergence. To verify the power of the scheme and to study its effectiveness for a variety of conditions, the simulations were done for various values for p, the probability of the Oracle correctly providing the feedback, and for various values of the resolution parameter, N.

In each case, the value of the parameter λ^* was assumed to be unknown to the LM. The experimental results obtained are truly conclusive. Although numerous experiments have been conducted, in the interest of brevity we shall first merely report the results obtained for one set of experiments involving the unknown parameter $\lambda^* = 0.9123$, which was the benchmark environment reported in the prior art [36]. However, we shall augment these with new results in which we are switching to a diametrically opposite environment, i.e., one in which $\lambda^* = 1 - 0.9123$. The results clearly demonstrate the power of our present strategy. We shall namely also show the results when λ^* is 0.22, 0.78, 0.35, and 0.07.

[13]In practice, as shown in the papers [41] and [42], this is really not an issue if we permit the LA parameter to be small enough, and also let the duration of the epoch to be sufficiently large.

TABLE II
Value of $E[\lambda(\infty)]$ for Different Values of p and Various Resolutions, When the Value of λ^* Is 0.9123

$log_2(N)$	$p = 0.7$	$p = 0.85$	$p = 0.95$
2	0.72790	0.82552	0.86637
3	0.82995	0.92040	0.93377
4	0.86781	0.90972	0.90763
5	0.89240	0.91925	0.92108
6	0.90565	0.91508	0.91444
7	0.90697	0.91111	0.91035
8	0.90826	0.91188	0.91202
9	0.91120	0.91284	0.91302
10	0.91164	0.91265	0.91262
11	0.91219	0.91244	0.91237
12	0.91211	0.91226	0.91223

A. Empirical Verification of Optimality of HHSL Solution

In Table II we have recorded an empirical estimate of the true value of $E[\lambda(\infty)]$ for various values of p and the tree depth $D = log_2(N)$ (i.e, resolution $N = 2^D$) when the value of λ^* was 0.9123. The values of p were 0.7, 0.85, and 0.95. This estimate of the value of $E[\lambda(\infty)]$ was obtained using simulation by running the learning scheme for 10^7 iterations.[14]

The reader should also observe that by invoking a significant number of iterations (as large as 10^7), the difference between the analytical and experimental values of $E[\lambda(\infty)]$ is unobservable. In the interest of simplicity, in this section, we therefore kindly request the reader to permit us to marginally abuse the notation and to refer to the experimental estimate of $E[\lambda(\infty)]$ as the value of $E[\lambda(\infty)]$ itself.

In every case the convergence of $E[\lambda(\infty)]$ was remarkable. For example, when p was as low as 0.7 and D was equal to 2, the value of $E[\lambda(\infty)]$ was as high as 0.727. It increased to 0.867 when $D = 4$ ($N = 16$) and came to within 0.5% of the true value. The results are more spectacular for more informative Oracles, i.e., for larger values of p. Thus, when p is 0.95, the value of $E[\lambda(\infty)]$ was as high as 0.866. The final terminal value when $D = 12$ represented an error less than 0.0001% . The power of the scheme is obvious!

The optimality property was empirically confirmed through the simulation, independent of whether the value of p was as low as 0.7 or as high as 0.95, because $E[\lambda(\infty)]$ indefinitely approaches the optimal value of λ^* as we increased the resolution. Note that for a depth D which equals 12, the final terminal value represented an error less than 0.0005% for all the values of p, i.e., $p = 0.7$, $p = 0.85$, and $p = 0.95$.

A similar plot of the asymptotic value of $\mathrm{E}[\lambda(\infty)]$ as a function of p is given in Fig. 3 for various resolutions, N. The experiment demonstrates how $\mathrm{E}[\lambda(\infty)]$ will change as the

[14]The reader should note that one could alternatively manually compute the stationary distribution equation by matrix inversion or by solving the system of linear equations [given by (2)] without even doing the simulations. We have opted to the latter because of the dimensionality of the transition matrices.

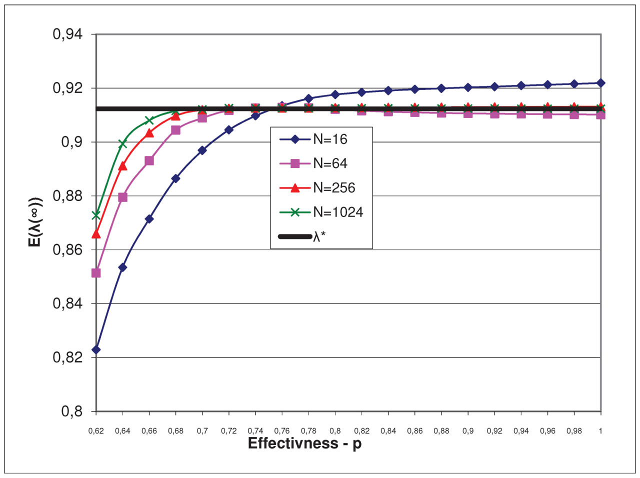


Fig. 3. Plot of the asymptotic value of E[$\lambda(\infty)$] with the effectiveness of the environment, p, for various values of the resolution parameter N, when λ^* is 0.9123.

effectiveness,[15] p, of the environment varied from $p = 0.62$ to $p = 1$.

B. Comparison Under Dynamic Environments

We now report the results of the second set of experiments in which we have attempted to catalogue the convergence of $E[\lambda(n)]$ with time, n.

In order to obtain an understanding as to how the scheme converged with time, various simulations were conducted to evaluate the performance of the algorithm under a variety of constraints. In each simulation, 1000 parallel experiments were conducted so that an accurate ensemble average of the results could be obtained. Although numerous experiments have been conducted, in the interest of brevity we shall merely report the results obtained for a single set of experiments involving the unknown parameter λ^* switching periodically between the values 0.9123 and $1 - 0.9123$. We also compared our results to the algorithm presented in [36].

In addition to comparing our new scheme to the original SPL approach, we also compared our work to the more-recent technique that involved Window-based estimation for nonstationary environments. Informally speaking, the estimate obtained by such a Window-based approach is merely the average of the time estimates of the SPL over a fixed-lengthed time window. We have used a simple rule of thumb in all our experiments, i.e., we have chosen the size of the window to be 1/10 of the length of the cycle that we are tracking, i.e, 1/10 of the the periodicity with which the environment switches.[16]

In every single experimental setting, we confirmed that our HSSL solution outperformed both the original SPL solution and the Window-based estimation solution. It also learned the value of λ^* much faster. The experimental results obtained are again conclusive.

[15] $p = 0.62$ is an approximation of the conjugate of the golden ratio.

[16]The reader must observe that in doing this, we have given the Window-based approach the advantage of having *some* knowledge of the periodicity of the switching environment!

Since there is no *a priori* information about the value of λ^*, at time instant 0, we initialized the LM of the original SPL scheme to the position $\frac{N}{2}$ (N is assumed to be even), while the initial position of the LM for the HSSL algorithm was merely the root node of the tree. In order to understand the effect of the resolution on the rate of convergence, we report the number of iterations required for the ensemble average to reach a value that is 95% close to the optimal value λ^*.

In our simulations, to obtain a 95% confidence level, we observed that the largest confidence interval was of the order of 0.06. This implies that the simulation results have also a very small variance. Since this confidence interval is too small to be visualized, we have opted to not include in the following graphs. Instead, we have rather chosen to report the instantaneous variances of the ensemble. These variance dynamics are only submitted for the experiments corresponding to Figs. 4 and 6, and are omitted for the rest of the experiments in the interest of space.

In the first set of experiments, we fixed p to be 0.8. The plots of the corresponding results are shown in Figs. 4 and 6. In Fig. 4, the resolution N was equal to 256 while λ^* switched every 400^{th} iteration. In Fig. 6, the resolution N was equal to be $1,024$ while λ^* switched every $1,500^{th}$ iterations.

From Fig. 4, where we have recorded the results of the first 400 iterations, we experimentally found that it took only 30 time instants for the HSSL solution to reach 95% of λ^*, while the original SPL solution and the Window-based estimate solution required 180 and 200 iterations, respectively. Therefore, in the very first window itself, the HSSL approach provided a superiority of an order of magnitude (6 times faster) than the original SPL solution. After the first environment switch, i.e., between time instants 400 and 800, we observed that the convergence speed of both algorithms decreased slightly. In fact, 95% of λ^* was attained within 45 iterations in the case of the HSSL paradigm, while the original SPL solution took 350 iterations. Comparing the results of the first 400 iterations with that of the subsequent windows, we conclude that although the final steady-state probabilities are independent of the starting state, in reality, the time that the LM takes to converge to λ^* is dependent on the location of the starting point. From this perspective, while the starting state of LM in the first window is $\frac{N}{2}$, in the second window, the starting point is the state to which it has converged in the first window. We believe that this is the explanation for the latter decrease in the convergence speeds.

Fig. 5 illustrates the evolution, over time, of the variance of the ensemble corresponding to the experimental settings of Fig. 4. The reader should note that the latter quantity is actually an estimate of $Var(\lambda(n))$. It is also pertinent to mention the following remarks. From the figure, we observe that during environment switches (or more precisely, at the time instances immediately subsequent to an environment switch), the variance of ensemble for the HSSL solution increased compared to the variance of both the SPL and Window-based estimate. One should also observe the peaks of the variances in Fig. 5 around time instances 400, 800, and $1,200$. We believe that this increase merely reflects the large steps that the LM of the HSSL takes to adapt to the change

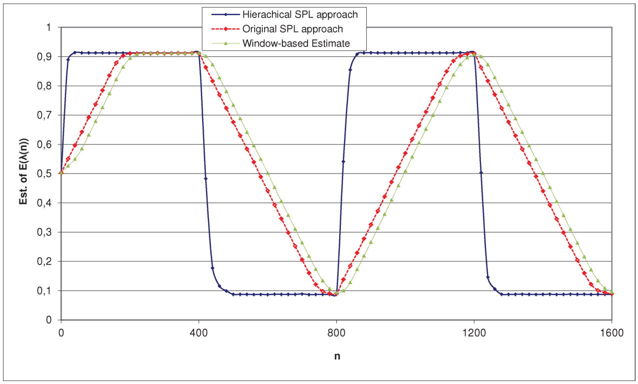


Fig. 4. Case where λ* switches between the values 0.9123 and 1.0–0.9123 every 400th iteration for $p = 0.8$ and $N = 256$.

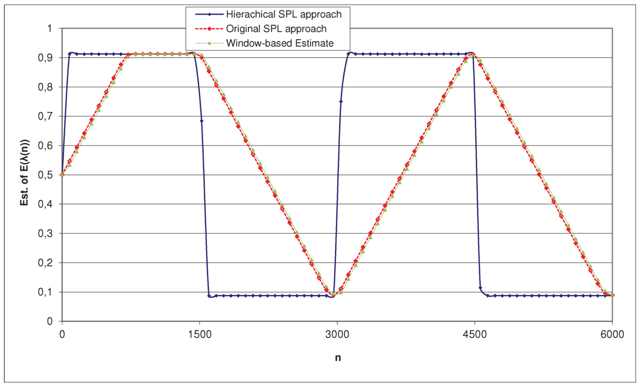


Fig. 6. Case where λ* switches between the values 0.9123 and 1.0–0.9123 every 1500th iterations for $p = 0.8$ and $N = 1024$.

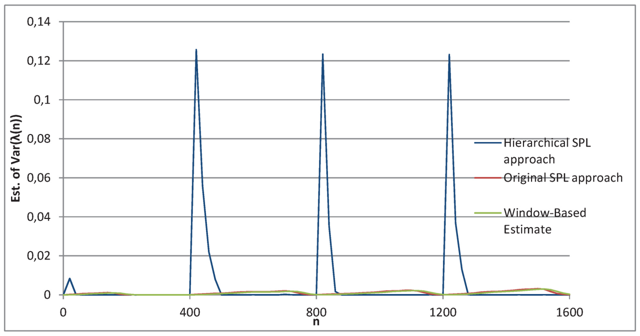


Fig. 5. Estimate of the instantaneous variance when λ* switches between the values 0.9123 and 1.0–0.9123 every 400th iteration for $p = 0.8$ and $N = 256$.

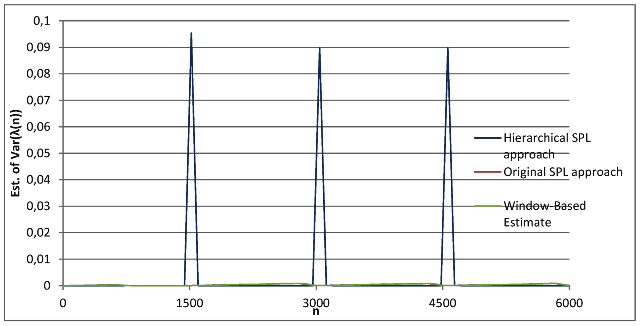


Fig. 7. Estimate of the instantaneous variance when λ* switches between the values 0.9123 and 1.0–0.9123 every 1500th iteration for $p = 0.8$ and $N = 1024$.

in the value of λ*, which is really, an intuitively appealing result! For example, at the time instant 410 (following the switch at 400), the variance of the ensemble for the HSSL is 0.12572 while the SPL and Window-based estimate have respective variance $2.3E - 4$ and $1.56E - 5$. Surprisingly though, these are the only time instances where the HSSL has higher variance than the SPL and the Window-based estimate. In fact, except for the time instances immediately subsequent to an environment switch, the HSSL was found to have a lower variance for the ensemble. For example, at the time instant 180, we experimentally recorded that the HSSL possessed a very low variance, i.e., $5.46E - 06$, which is much smaller than the variance of the SPL $2.96E - 04$. To quote another similar example, we mention that at the time instant 1,580, the variance of the ensemble for the HSSL was as low as $9.03E - 07$ while the corresponding variance of the SPL was many orders of magnitude larger, i.e., $2.5E-04$. This confirms that these peaks in the variance reflect the large steps that the HSSL has to take in order to adapt to an environment switch, while the low variance reflects the fact that the HSSL is approaching λ* and that the learning mechanism is working toward exhibiting less fluctuations by performing top-down transitions in the direction of the target leaf node.

A general remark that we should make regarding comparing the SPL to the Window-based estimate, is that most of the time, the Window-based estimate had a lower instantaneous variance than the SPL in all the experiments that we did. We believe that this is due to the fact that the Window-based estimate averages out the fluctuations of the estimates obtained by the SPL, and that it therefore achieves a more accurate estimate than the latter. However, on the other hand, we also observed that the Window-based estimate it is less adaptive to changes in λ* when compared to the SPL.

In Fig. 6, we increased the resolution N to 1,024. As in the previous case, from Fig. 6, we observed that, in the first 1500 iterations, it took approximately 50 iterations for our HSSL solution to reach 95% of the optimal value λ*, while the original SPL solution required 680 iterations. After the first environment switch, i.e, between time instants 1500 and 3000, we observed that the convergence speed again decreased. In fact, it took approximately 75 iterations for our HSSL solution to reach 95% of the optimal value λ*, while the original SPL solution required 1380 iterations. Again, in these settings, the HSSL approach provided an order of magnitude (18 times) faster convergence than the original SPL solution. This, we believe, is impressive. We further remark that, as we increased the resolution N from 256 (see Fig. 4) to 1024 (see Fig. 6) for the same value of $p = 0.8$, the convergence speed of the original SPL solution was significantly reduced while the speed of the HSSL was less affected by this increase in the resolution.

In the second set of experiments, we report the results for the values $p = 0.95$. The plots of the corresponding results are shown in Figs. 8 and 9. In Fig. 8, the resolution N was equal to 256 while λ* switched every 300th iteration. In Fig. 9, the resolution N was equal to 1024 while λ* switched every 1200th iteration.

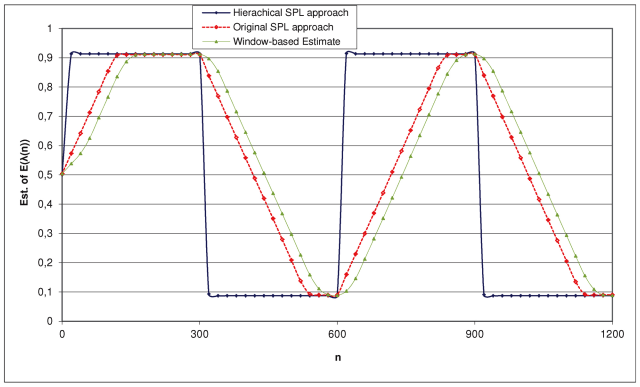


Fig. 8. Case where λ* switches between the values 0.9123 and (1–0.9123) every 300th iteration for $p = 0.95$ and $N = 256$.

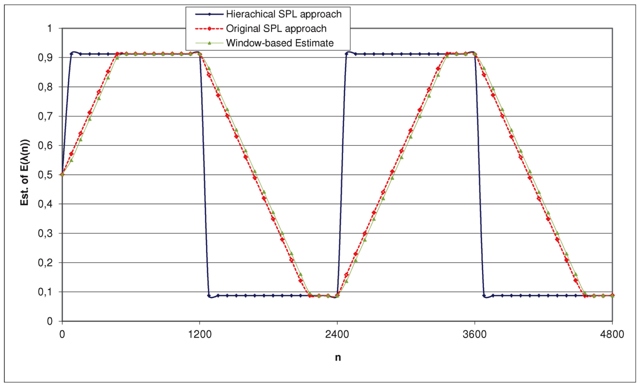


Fig. 9. Case where λ* switches between the values 0.9123 and (1–0.9123) every 1200th iteration for $p = 0.95$ and $N = 1024$.

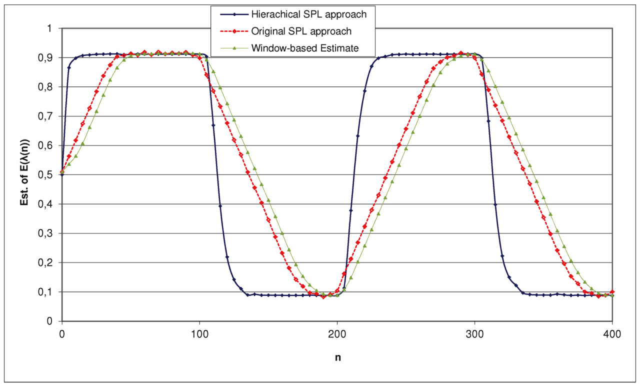


Fig. 10. Case where λ* switches between the values 0.9123 and (1–0.9123) every 100th iteration for $p = 0.85$ and $N = 64$.

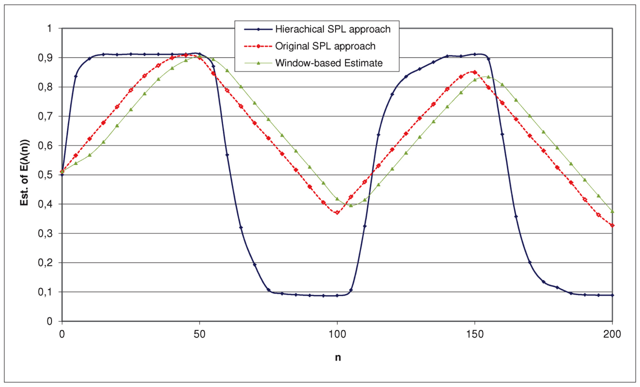


Fig. 11. Case where λ* switches between the values 0.9123 and (1–0.9123) every 50th iteration for $p = 0.85$ and $N = 64$.

From Fig. 8, in the first window of 300 iterations, we experimentally found that it took only 14 time instants for the HSSL solution to reach 95% of λ^*, while the original SPL solution needed 120 iterations. In the second window between time instants 300 and 600, 95% of λ^* was reached within 19 iterations in the case of the HSSL, while the original SPL solution took 220 iterations. Again, we see the HSSL's superiority.

In Fig. 9, we changed the resolution N to 1024. Similar to the behavior shown in Fig. 8, one can observe from Fig. 9 that in the first 1200 iterations, it took approximately ten iterations for our HSSL solution to reach 95% of the optimal value λ^*, while the original SPL solution required 510 iterations, a superiority of a factor of about 50. After the first environment switch, i.e, between time instants 1200 and 2400, we again observed that the convergence speed decreases. In fact, it took approximately 14 iterations for our HSSL solution to reach 95% of the optimal value λ^*, while the original SPL solution took 920 iterations. Hence, in this window, the HSSL approach is approximately 65 faster than the original SPL solution!

In addition, as we increased the resolution N from 256 (see Fig. 8) to 1024 (see Fig. 9) for the same value of $p = 0.95$, the convergence speed of the original SPL solution was more drastically affected than that of the HSSL approach.

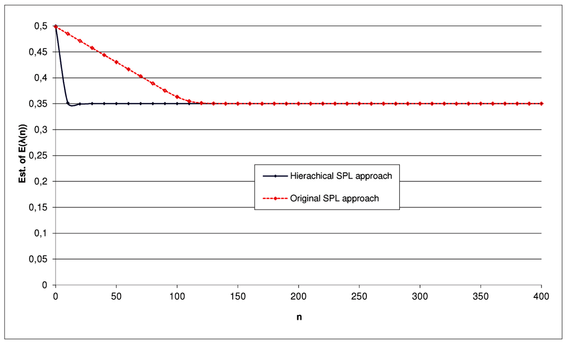
As can be expected, the results of the HSSL solution are more spectacular for more informative Oracles, i.e., for larger values of p. In fact, as we increased the effectiveness of the environment p from $p = 0.85$ to $p = 0.95$ the convergence rate of the HSSL increased even more.

In this third set of experience, we fixed the resolution $N = 64$ and the effectiveness $p = 0.85$ while we varied the periodicity with which λ^* switched. Fig. 10 depicts the case where the switch occurred every 100th iteration, while Fig. 11 illustrates the case where the switch occurred every 50^{th} iteration.

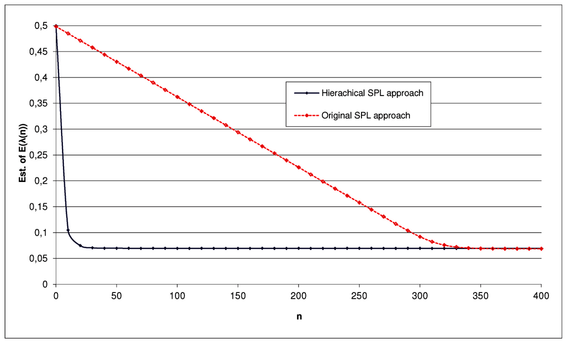
From Fig. 11, we observe that the original SPL solution was able to converge to $E[\lambda(\infty)]$ in the first window, but it was thereafter quite handicapped with regard to adapting to the rapid switches in the nonstationary environment. Since the changes in the value of λ^* happened at a faster time scale than the time required by the LM to converge to $E[\lambda(\infty)]$, this scenario demonstrates the weakness of the original SPL. Such observations are typical. The reader should note that in the first set and second set of experiments (Figs. 4, 6, 8, and 9), we deliberately changed λ^* at a slower time scale so that we could observe the convergence of the original SPL in the different windows, and we could thus quantify the required number of iterations needed for it to converge.

C. *Effect of Choice of* λ^* *on Convergence Speed*

The intent of this section is to demonstrate the power of the HSSL when λ^* is changed. Based on the theoretical results, we do not expect any change of behavior for other values of



(a)



(b)

Fig. 12. Effect of changing the value of λ^* on the convergence speed of the HSSL and the original SPL in the cases where (a) $\lambda^* = 0.35$. (b) $\lambda^* = 0.07$.

λ^*, and this section empirically shows that the performance of the HSSL is in line with our expectations. Indeed, the convergence of the scheme is not effected by the value of λ^*; it can efficiently locate λ^* whatever its value is.

In this experiment, our aim was to investigate the effect of choosing a value of λ^* different from the value that is commonly used in the literature, i.e., 0.9123. Thus, in this experiment, we chose four different values for λ^*, namely 0.22, 0.78, 0.35, and 0.07. For both these experiments, we fixed the resolution to be 512 and p to 0.85. In all brevity, we observed that the convergence speed for the HSSL did not depend on λ^* but rather only on the number of levels in the tree.

These findings bear similarity with the classical results dealing with the access time complexity of ordered records using a sequential access policy and a tree-based indexing policy [23]. The only difference between these and our current scenario is that we are dealing with stochastic responses from the environment. When it concerns the SPL, we observe from Fig. 12(a) and (b) that the more distant λ^* is from the starting point of the search 0.5, the more time it takes for the scheme to converge. In fact, it take more time instances for the SPL to converge to the optimal estimate when $\lambda^* = 0.07$ than when $\lambda^* = 0.35$ since the distance between 0.5 and 0.07 is larger than the distance between 0.5 and 0.35. However, when it comes to the HSSL, we empirically verify from Fig. 12(a) and (b) that the convergence speed is the same for both values of λ^*. This is also intuitively appealing.

Fig. 13 illustrates an intuitive property of both the HSSL and SPL. In effect, the convergence speed is identical for any two values λ^* and $1-\lambda^*$ because of the fact that the two points are diametrically symmetric around the initial point 0.5. We, therefore, conducted the experiments only for the case when the values were smaller than 0.5. The experimental results for values of λ^* larger than 0.5 can simply be deduced by invoking the property of symmetry, as we did in the case of Fig. 13.

D. Comparison With Tertiary Search

In this final experiment, our aim was to compare the HSSL with results of the tertiary search strategy reported in [41]. In order to make the comparison meaningful (since the HSSL is not working with a parallelized scheme), we fixed the number of epochs of the tertiary search [41] to be equal to the number of levels in the tree structure of the HSSL.

Each epoch consisted of 50 iterations and the internal parameter θ of the linear reward inaction (LRI) automata for the tertiary search was fixed to be 0.8, a value that is known to yield a good accuracy for the search. We fixed $\lambda^* = 0.9123$ and we varied the efficiency of the environment as in the experiments reported in [41] using the values of $p = 0.7$, $p = 0.85$ and $p = 0.95$. The results obtained are depicted in Fig. 14. Based on our rather extensive testing from which Fig. 14(a)–(c) display representative results, we remark that the HSSL is always faster than the tertiary search. The main disadvantage of the tertiary search is that it runs in epochs, implying that we can only obtain an estimate of λ^* at the end of the epoch. Thus, a more appropriate way to have plotted the the results of the tertiary search for different time instances would have been to use a staircase function with constant values during the entirety of every single epoch (i.e, for an epoch of length 50 time instants, the estimate of λ^* is constant between time instant 0 and 49 and so on). Such a display would have demonstrated the even greater superiority of the HSSL.

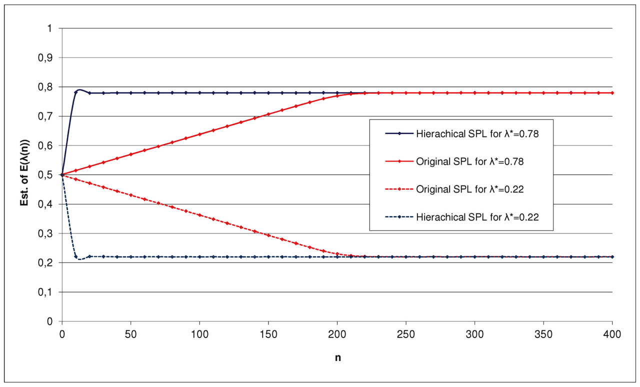
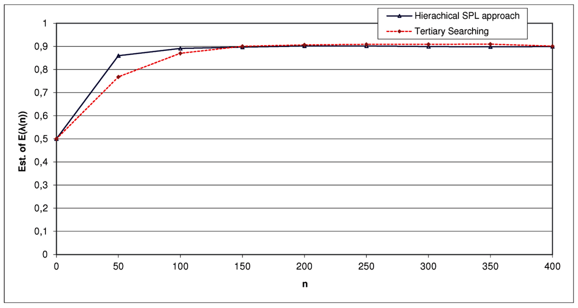


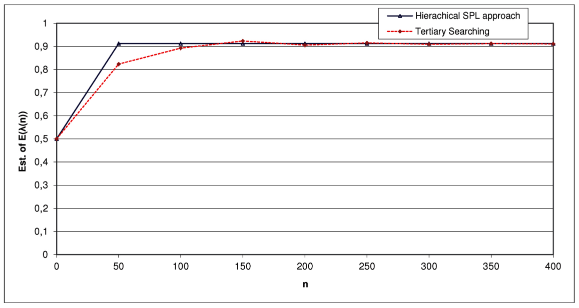
Fig. 13. Effect of changing the value of λ^* on the convergence speed of the HSSL and the original SPL in the case of two symmetrical values of λ^* around 0.5, namely, (a) $\lambda^* = 0.22$. (b) $\lambda^* = 0.78$.

E. Measuring Rate of Convergence Using Hit Times

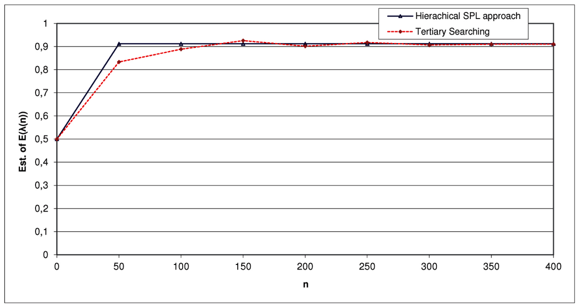
An alternative metric to quantify the rate of convergence for the HSSL is to measure the hit time of the first visit to



(a)



(b)



(c)

Fig. 14. Comparing the HSSL approach with the tertiary searching method for different experimental settings. (a) $p = 0.7$. (b) $p = 0.85$. (c) $p = 0.95$.

the target node.[17] We now report the average hit time for an ensemble of experiments as a measure of the scheme's rate of convergence. Since the SPL-based approach is also founded on a Markov chain, it is also possible to quantify the hit time for this approach. The hit time of the original SPL solution is defined as the first visit to the node whose index s is such as $s/N \leq \lambda < (s+1)/N$.

Note that the hit time is a random variable whose distribution is unknown. Hence, we can calculate the confidence interval using Chebyshev's inequality, which states that at least $100(1-1/k^2)\%$ of the probability mass lies around the mean μ and in the interval $\mu \pm k\sigma$ where σ is standard deviation, and k is a user-defined parameter that controls the size of the confidence interval. For the 95% confidence interval, k is approximately equal to 4.47. Using this formulation, the confidence interval is given by: $[\mu - 4.47\sigma, \mu + 4.47\sigma]$. In our experiments, as in the previous cases, we fixed the resolution N to be 256, and λ^* to 0.9123 and examined the evolution of the average hit time as a function of the parameter, p. Again, the number of experiments in the ensemble was set to be 1000, and the initial state of the LM corresponded to an estimate equal to 0.5 (i.e., the initial state for the LM modeling the HSSL was the root node, while initial state for LM modeling the original SPL was $N/2$).

As expected, when $p = 1$, the search was deterministic and the hit time for the HSSL corresponded to the depth of the tree, namely 8, while the hit time for the SPL was 105 since $0.5 + 105/256 \leq \lambda^* < 0.5 + 106/256$. The reader should note that, given any value of p, these two values, 8 and 105, present

[17]We are grateful to the anonymous referee who suggested this comparison metric as well as that of deriving the confidence intervals in this section using Chebyshev's inequality.

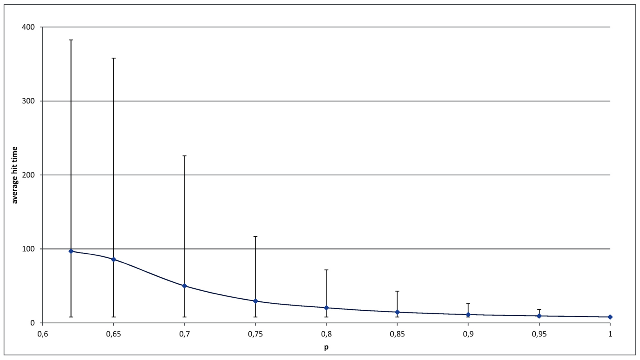


Fig. 15. Average hit time of the HSSL as a function of p with a corresponding 95% confidence interval for $\lambda^* = 0.9123$ and $N = 256$.

a lower bound on the respective hit times of the HSSL and SPL approaches. The knowledge of these lower bounds for the hit times permitted us to reduce the size of the confidence interval. In fact, the confidence interval can be rewritten as $[max(\mu - 4.47\sigma, 8), \mu + 4.47\sigma]$ for the HSSL and $[max(\mu - 4.47\sigma, 105), \mu + 4.47\sigma]$ for the original SPL. Fig. 15 reports the average hit time of the HSSL approach, while Fig. 16 reports the results for the SPL solution.

From both these figures, we see that the HSSL had a significantly superior rate of convergence. For example, when $p = 0.8$, the average hit time of the HSSL was 20.4 while the corresponding quantity for the original SPL was 174.16. For $p = 0.65$, the average hit time for the HSSL was 83 while the hit time for the original SPL was 348.5. We also remark that for both schemes, the confidence interval increases as p decreases. In addition, one can observe that the confidence interval of the SPL is much larger than the confidence interval

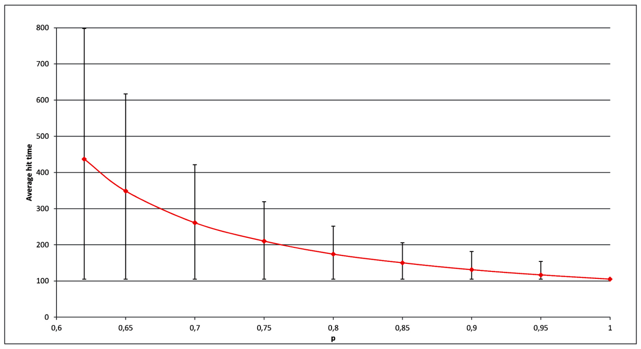


Fig. 16. Average hit time of the SPL as a function of p with a corresponding 95% confidence interval for $\lambda^* = 0.9123$ and $N = 256$.

of the HSSL for different values of p which, in turn, reflects a lower variance for the hit time of the HSSL. For example, when $p = 0.62$, the confidence interval of the HSSL was approximately $[8, 360]$, while the confidence interval for the original SPL was approximately $[105, 797]$.

VII. Conclusion

The HSSL problem involves a Learning Machine (LM) that attempts to learn a parameter λ^* within a closed interval. For each guess, the environment essentially informs the mechanism with three possible responses, each possibly erroneous (i.e., with probability p), which way it should move to reach the unknown point. Thus, unlike the deterministic expandable binary search strategy of [9], we consider the fascinating case where the response of the environment is assumed to be faulty. We have presented a solution that involves discretizing the space into a complete binary tree and performing a controlled random walk on this space. The solution we have presented has been both analytically analyzed and simulated, with extremely interesting results. Apart from formally analyzing the HSSL algorithm, we have also experimentally demonstrated its superiority over the state-of-the-art. From this perspective, our approach is shown to provide orders of magnitude faster convergence than the traditional SPL solution [36] when tested in nonstationary environments where λ^* changes over time.

To summarize, this paper presents a set of novel contributions[18] listed below.

1) With regard to the design of discretized parameter schemes, we submit that a fundamental contribution of this paper is the manner in which we have designed the discretized search space, by structuring it as a binary tree. Our SPL solution invokes a merging of two completely disjoint fields in computer science, those of stochastic optimization and data structures.
2) We have also proposed a new scheme for operating on this discretized space. Traditional approaches for discretization work by restricting the corresponding parameter to be one of finite number of values in the interval $[0, 1]$, and by then performing a uni-dimensional controlled random walk on the discretized space, where the transitions only occur between neighbor nodes, i.e, to the left or to the right. We propose that the scheme be permitted to migrate (i.e., resort to an enhanced exploration domain) to nonneighbor values in the search space, where the latter are governed by the underlying data structure.
3) With regard to discretization, we have also proposed a new concept of resolution. In traditional discrete schemes, the convergence speed is decreased as the resolution N increases, while the accuracy increases, and vice versa. Instead, as per our new philosophy in which the parameter space is structured as a binary tree, briefly, we associate a resolution to each level of the tree, and this resolution becomes finer at deeper levels of the tree. Empirical results show that there is a significant merit in our proposed tree-based discretization philosophy when it is compared to the traditional discretization model.
4) It goes without saying that the paper presents a significant contribution to the set of solutions to the SPL problem. Extensive simulations results confirm that our scheme outperforms the state-of-the art schemes by orders of magnitude. In addition, simulations results show that our scheme possesses an excellent ability to cope with nonstationary environments.
5) A rather fundamental contribution of this paper is the analytical solution of the Markov chain for this novel hierarchical learning paradigm. The earlier solutions to traditional discretized learning paradigms involve the closed-form analysis of uni-dimensional random walks, where the transitions are primarily only to neighbor states. In this paper, the random walk is performed, instead, on a balanced tree, which renders the problem more complex. We report the first analytical results for the HSSL and prove that the HSSL is asymptotically optimal. We believe that the analysis of the scheme we report here is a contribution in its own right, to the field of LA and to the theory of Markov chains.
6) Finally, and most importantly, the results obtained are extremely fascinating, as this is the first reported use of time reversibility in the analysis of stochastic learning. The LA extensions of the scheme are currently being investigated, and we believe, that they hold vast potential. It would truly be very intriguing if we could design large-step (as opposed to small-step) ϵ-optimal LA based on these principles.

As a future work, we propose to investigate the use of the HSSL solution to solve practical stochastic optimization problems. The generalization of the HSSL procedure proposed in this paper to a search space structured as a general tree (i.e, not necessarily binary) is an open research question. Moreover, we are currently investigating the potential of making use of the discretization philosophy introduced here to improve the convergence speed of LA algorithms, and in analyzing the HSSL scheme when operating in the meta-level nonstationarity environment studied in [39]. We would also like to compare the HSSL with autoregressive parameter identification filters as described in [25].

[18]We are grateful to one of the referees of the paper who recommended cataloging the list of contributions in this concluding section.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees whose suggestions greatly improved the quality of this paper.

REFERENCES

[1] S. Afshar, M. Mosleh, and M. Kheyrandish, "Presenting a new multiclass classifier based on learning automata," *Neurocomputing*, vol. 104, no. 0, pp. 97–104, 2013.

[2] M. Agache and B. J. Oommen, "Generalized pursuit learning schemes: New families of continuous and discretized learning automata," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 6, pp. 738–749, Dec. 2002.

[3] N. Baba and Y. Mogami, "A new learning algorithm for the hierarchical structure learning automata operating in the nonstationary s-model random environment," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 6, pp. 750–758, Dec. 2002.

[4] N. Baba and Y. Mogami, "A consideration on the learning behaviors of the hierarchical structure learning automata operating in the non-stationary multiteacher environment: A basic research to realize an effective utilization of artificial neural networks in the nonstationary environment," in *Proc. 34th Annu. Conf. ISAGA*, 2003, pp. 1021–1030.

[5] N. Baba and Y. Mogami, "Learning behaviors of the hierarchical structure stochastic automata operating in the nonstationary multiteacher environment," in *Proc. KES*, Sep. 2005, pp. 624–634.

[6] R. A. Baeza-Yates, J. C. Culberson, and G. J. E. Rawlins, "Searching with uncertainty," in *Proc. SWAT*, 1998, pp. 176–189.

[7] R. Baeza-Yates and R. Schott, "Parallel searching in the plane," *Comput. Geom. Theory Appl.*, vol. 5, pp. 143–154, Oct. 1995.

[8] T. Ben-Zvi and J. V. Nickerson, "Decision analysis: Environmental learning automata for sensor placement," *IEEE Sens. J.*, vol. 11, no. 5, pp. 1206–1207, May 2011.

[9] J. L. Bentley and A. C.-C. Yao, "An almost optimal algorithm for unbounded searching," *Inf. Process. Lett.*, vol. 5, no. 3, pp. 82–87, 1976.

[10] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd ed. Belmont, MA, USA: Athena Scientific, 2000.

[11] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag, 2006.

[12] O.-C. Granmo and B. J. Oommen, "Solving stochastic nonlinear resource allocation problems using a hierarchy of twofold resource allocation automata," *IEEE Trans. Comput.*, vol. 59, no. 4, pp. 545–560, Apr. 2010.

[13] O.-C. Granmo and B. J. Oommen, "Learning automata-based solutions to the optimal web polling problem modelled as a nonlinear fractional knapsack problem," *Eng. Appl. AI*, vol. 24, no. 7, pp. 1238–1251, 2011.

[14] D. Graupe, *Principles of Artificial Neural Networks*, 2nd ed. River Edge, NJ, USA: World Scientific Publishing, 2007.

[15] M. A. Hossain, J. Parra, P. K. Atrey, and A. E. Saddik, "A framework for human-centered provisioning of ambient media services," *Multimedia Tools Applicat.*, vol. 44, pp. 407–431, 2009.

[16] D.-S. Huang and W. Jiang, "A general CPL-ADS methodology for fixing dynamic parameters in dual environments," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 5, pp. 1489–1500, Oct. 2012.

[17] S. Karlin and H. M. Taylot, *A First Course in Stochastic Processes*. San Diego, CA, USA: Academic Press, 1975.

[18] F. P. Kelly, *Reversibility and Stochastic Networks (Wiley Series in Probability and Mathematical Statistics Tracts on Probability and Statistics)*. New York, NY, USA: Wiley, 1987.

[19] E. E. Kpamegan and N. Flournoy, "Up-and-down designs for selecting the dose with maximum success probability," *Seq. Anal.*, vol. 27, no. 1, pp. 78–96, 2008.

[20] S. Lakshmivarahan, *Learning Algorithms Theory and Applications*. New York, NY, USA: Springer-Verlag, 1981.

[21] J. K. Lanctôt and B. J. Oommen, "Discretized estimator learning automata," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 6, pp. 1473–1483, Nov./Dec. 1992.

[22] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[23] A. Levitin, *Introduction to the Design & Analysis of Algorithms*. Reading, MA, USA: Addison Wesley, 2011.

[24] M. Livio, *The Golden Ratio: The Story of Phi, the World's Most Astonishing Number*. Portland, OR, USA: Broadway Books, 2003.

[25] L. Ljung, *System Identification: Theory for the User*. Upper Saddle River, NJ, USA: Prentice-Hall, 1986.

[26] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[27] S. Misra, P. V. Krishna, V. Saritha, and M. S. Obaidat, "Learning automata as a utility for power management in smart grids," *IEEE Commun. Mag.*, vol. 51, no. 1, pp. 98–104, Jan. 2013.

[28] B. T. Mitchell and D. I. Kountanis, "Reorganization scheme for a hierarchical system of learning automata," *IEEE Trans. Syst., Man, Cybern.*, vol. SMNC-14, no. 2, pp. 328–334, Mar./Apr. 1984.

[29] K. Najim and A. S. Poznyak, *Learning Automata: Theory and Applications*. New York, NY, USA: Pergamon Press, 1994.

[30] K. S. Narendra and K. Parthasarathy, "Learning automata approach to hierarchical multiobjective analysis," Dept. Electr. Eng., Yale University, New Haven, CT, USA, Tech. Rep. 8811, 1988.

[31] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.

[32] P. Nicopolitidis, V. Kakali, G. Papadimitriou, and A. Pomportsis, "On performance improvement of wireless push systems via smart antennas," *IEEE Trans. Commun.*, vol. 60, no. 2, pp. 312–316, Feb. 2012.

[33] P. Nicopolitidis, G. I. Papadimitriou, A. S. Pomportsis, P. Sarigiannidis, and M. S. Obaidat, "Adaptive wireless networks using learning automata," *IEEE Wireless Commun.*, vol. 18, no. 2, pp. 75–81, Apr. 2011.

[34] M. S. Obaidat, G. I. Papadimitriou, and A. S. Pomportsis, "Guest editorial—Learning automata: Theory, paradigms, and applications," *IEEE Trans. Systems, Man, Cybern. B, Cybern.*, vol. 32, no. 6, pp. 706–709, Dec. 2002.

[35] B. J. Oommen, "Absorbing and ergodic discretized two-action learning automata," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-16, no. 2, pp. 282–293, Mar./Apr. 1986.

[36] B. J. Oommen, "Stochastic searching on the line and its applications to parameter learning in nonlinear optimization," *IEEE Trans. Syst., Man Cybern.*, vol. SMC-27B, no. 4, pp. 733–739, Aug. 1997.

[37] B. J. Oommen, O-C. Granmo, and Z. Liang, "A novel stochastic learning-enhanced multidimensional scaling technique applicable for word-of-mouth discussions," in *Proc. IEA/AIE*, Jun. 2009, pp. 317–322.

[38] B. J. Oommen and E. Hansen, "The asymptotic optimality of discretized linear reward-inaction learning automata," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-14, no. 3, pp. 542–545, May/Jun. 1986.

[39] B. J. Oommen, S.-W. Kim, M. T. Samuel, and O.-C. Granmo, "A solution to the stochastic point location problem in metalevel nonstationary environments," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 2, pp. 466–476, Apr. 2008.

[40] B. J. Oommen and J. K. Lanctôt, "Discretized pursuit learning automata," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-20, no. 4, pp. 931–938, Jul./Aug. 1990.

[41] B. J. Oommen and G. Raghunath, "Automata learning and intelligent tertiary searching for stochastic point location," *IEEE Trans. Syst., Man, Cybern.*, vol. 28, no. 6, pp. 947–954, Dec. 1998.

[42] B. J. Oommen, G. Raghunath, and B. Kuipers, "Parameter learning from stochastic teachers and stochastic compulsive liars," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-36, no. 4, pp. 820–836, Aug. 2006.

[43] B. J. Oommen and M. Agache, "Continuous and discretized pursuit learning schemes: Various algorithms and their comparison," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 31, no. 3, pp. 277–287, Jun. 2001.

[44] B. J. Oommen and J. Dong, "Generalized swap-with-parent schemes for self-organizing sequential linear lists," in *Proc. ISAAC*, Dec. 1997, pp. 414–423.

[45] G. I. Papadimitriou, "Hierarchical discretized pursuit nonlinear learning automata with rapid convergence and high accuracy," *IEEE Trans. Knowl. Data Eng.*, vol. 6, no. 4, pp. 654–659, Aug. 1994.

[46] J. R. Parker, *Algorithms for Image Processing and Computer Vision*. New York, NY, USA: Wiley, 2010.

[47] T. Pavlidis, *Structural Pattern Recognition*. New York, NY, USA: Springer-Verlag, 1977.

[48] R. Phillips, *Pricing and Revenue Optimization (Stanford Business Books)*. Stanford, CA, USA: Stanford Univ. Press, 2005.

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

[50] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*. Cambridge, U.K.: Cambridge Univ. Press, 1986.

[51] S. M. Ross, *Introduction to Probability Models*. San Diego, CA, USA: Academic Press, 1980.

[52] G. Santharam, P. S. Sastry, and M. A. L. Thathachar, "Continuous action set learning automata for stochastic optimization," *J. Franklin Inst.*, vol. 331, no. 5, pp. 607–628, Sep. 1994.

[53] B. Siciliano and O. Khatib, Eds. *The Handbook of Robotics*. Berlin/Heidelberg, Germany: Springer, 2008.

[54] M. A. L. Thathachar and B. J. Oommen, “Discretized reward-inaction learning automata,” *J. Cybern. Inf. Sci.*, vol. 2, no. 1, pp. 24–29, 1979.
[55] M. A. L. Thathachar and K. R. Ramakrishnan, “A hierarchical system of learning automata,” *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, no. 3, pp. 236–241, Mar. 1981.
[56] M. A. L. Thathachar and P. S. Sastry, *Networks of Learning Automata: Techniques for Online Stochastic Optimization*. Boston, MA, USA: Kluwer Acadmic Publishers, 2004.
[57] O. Tilak, R. Martin, and S. Mukhopadhyay, “Decentralized indirect methods for learning automata games,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 5, pp. 1213–1223, Oct. 2011.
[58] J. A. Torkestani and M. R. Meybodi, “Finding minimum weight connected dominating set in stochastic graph based on learning automata,” *Inf. Sci.*, vol. 200, no. 0, pp. 57–77, 2012.
[59] P. D. Wasserman, *Neural Computing: Theory and Practice*. New York, NY, USA: Van Nostrand Reinhold, 1989.
[60] Y. Xu, J. Wang, Q. Wu, A. Anpalagan, and Y.-D. Yao, “Opportunistic spectrum access in unknown dynamic environment: A game-theoretic stochastic learning solution,” *IEEE Trans. Wireless Commun.*, vol. 11, no. 4, pp. 1380–1391, Apr. 2012.
[61] A. Yazidi, O.-C. Granmo, and B. J. Oommen, “Service selection in stochastic environments: A learning-automaton based solution,” *Appl. Intell.*, vol. 36, no. 3, pp. 617–637, 2012.
[62] P. Zhou, Y. Chang, and J. A. Copeland, “Reinforcement learning for repeated power control game in cognitive radio networks,” *IEEE J. Sel. Areas Commun.*, vol. 30, no. 1, pp. 54–69, Jan. 2012.

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

He is currently an Associate Professor with the Department of Computer Science, Oslo and Akershus University College of Applied Sciences, Oslo, Norway. He was a Researcher with Teknova AS, Grimstad. His current research interests include machine learning, learning automata, stochastic optimization, recommendation systems, pervasive computing, and the applications of these areas in industrial settings.

Ole-Christoffer Granmo was born in Porsgrunn, Norway. He received the M.Sc. and the Ph.D. degrees from the University of Oslo, Oslo, Norway, in 1999 and 2004, respectively.

He is currently a Professor with the Department of ICT, University of Agder, Grimstad, Norway. He is the author of over 70 refereed journal and conference publications. His current research interests include intelligent systems, stochastic modeling and inference, machine learning, pattern recognition, learning automata, distributed computing, and surveillance and monitoring.

B. John Oommen (F’03) was born in Coonoor, India, on September 9, 1953. He received the B.Tech. degree from the Indian Institute of Technology Madras, Chennai, India, in 1975, the M.E. degree from the Indian Institute of Science, Bangalore, India, in 1977, and the M.S. and Ph.D. degrees from Purdue University, West Lafayettte, IN, USA, in 1979 and 1982, respectively.

He joined the School of Computer Science, Carleton University, Ottawa, ON, Canada, in the 1981–82 academic year, where he is currently a Full Professor. He is the author of over 395 refereed journal and conference publications. His current research interests include automata learning, adaptive data structures, statistical and syntactic pattern recognition, stochastic algorithms and partitioning algorithms.

Dr. Oommen has been awarded the honorary rank of Chancellor’s Professor in July 2006, which is a lifetime award from Carleton University. He has also served on the Editorial Board of the IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS, AND PATTERN RECOGNITION. He is a fellow of the IAPR.

Morten Goodwin was born in Nøtterøy, Norway. He received the B.Sc. and M.Sc. degrees from Agder University College, Grimstad, Norway, 2005 and 2003, respectively, and the Ph.D. degree with the thesis “Towards Automated eGovernment Monitoring” from Aalborg University, Aalborg, Denmark, in 2011.

He is currently an Associate Professor with the Department of ICT, University of Agder, Grimstad. He has published in several high ranked scientific publications. His current research interests include data-mining, optimization, machine learning, and software development.