Solving Stochastic Root-Finding with Adaptive d-ary Search

Anis Yazidi
Dept. of Computer Science
University College of Oslo and Akershus
Oslo, Norway
Email: anis.yazidi@hioa.no

B. John Oommen
School of Computer Science
Carleton University, Ottawa, Canada
oommen.scs@carleton.ca

***Abstract*—The most fundamental problem encountered in the field of stochastic optimization, is the Stochastic Root Finding (SRF) problem where the task is to locate an unknown point x^* for which $g(x^*) = 0$ for a given function g that can only be observed in the presence of noise [13]. The vast majority of the state-of-the-art solutions to the SRF problem involve the theory of stochastic approximation. The premise of the latter family of algorithms is to operate by means of so-called "small-step" processes that explore the search space in a conservative manner. Using this paradigm, the point investigated at any time instant is in the proximity of the point investigated at the previous time instant, rendering the convergence towards the optimal point, x^*, to be sluggish. The unfortunate thing about such a search paradigm is that although $g()$ contains information using which large sections of the search space can be eliminated, this information is unutilized. This paper provides a pioneering and novel scheme to discover and utilize this information. Our solution recursively shrinks the search space by, at least, a factor of $\frac{2d}{3}$ at each epoch, where $d \geq 2$ is a user-defined parameter of the algorithm. This enhances the convergence significantly. Conceptually, this is achieved through a subtle re-formulation of SRF problem in terms of a continuous-space generalization of the Stochastic Point Location (SPL) problem originally proposed by Oommen in [8]. Our scheme is based, in part, on the Continuous Point Location with Adaptive d-ary Search (CPL–AdS), originally presented in [12]. The solution to the CPL–AdS [12], however, is not applicable in our particular domain because of the inherent asymmetry of the SRF problem. Our solution invokes a CPL–AdS-like solution to partition the search interval into d sub-intervals, evaluates the location of the unknown root x^* with respect to these sub-intervals using learning automata, and prunes the search space in each iteration by eliminating at least one partition. Our scheme, the CPL–AdS algorithm for SRF, denoted as SRF–AdS, is shown to converge to the unknown root x^* with an arbitrary large degree of accuracy, i.e., with a probability as close to unity as desired. Unlike the classical formulation of the SPL problem proposed by Oommen *et al* [8], [12], in our setting, the probability, p, of the "environment" suggesting an accurate response is non-constant. In fact, the latter probability depends of the point x being examined and the region that is a candidate to be pruned. The fact that p is not constant renders the analysis much more involved than in [12]. The decision rules for pruning are also different from those encountered when p is constant [12].**

Keywords: *Stochastic Root finding Problem, Stochastic Point Location Problem, Learning Automata*

B. J. Oommen is a *Chancellor's Professor*, a *Fellow: IEEE* and a *Fellow: IAPR*. The Author also holds an *Adjunct Professorship* with the Dept. of ICT, University of Agder, Norway. His work was partially supported by NSERC, the Natural Sciences and Engineering Research Council of Canada.

I. Introduction

Any optimization problem involves, in one way or another, the issue of solving for the "root" of a function because the maximum/minimum of a function occurs when its (partial) derivatives are zero. The problem is much more complex when the function whose root is sought for is stochastic, i.e., one does not have access to the function itself but only to its noisy/inexact evaluations. This naturally leads us to the so-called Stochastic Root Finding (SRF) problem, whose applications are all-pervasive in stochastic optimization. The state-of-the-art techniques for solving the SRF problem build on the well-established and pioneering Robbins-Monro algorithm [13] where the pioneers provided a recursive updating formulation using the theory of diminishing step sizes. More specifically, the form of the recursive update can be specified as:

$$x_{n+1} = x_n + a_n Y_n(x_n), \text{ with}$$

$$Y_n(x_n) = g(x_n) + w.\ g(.),$$

where x_n is the point sampled at the time instant n. In the above, $Y_n(x_n)$ denotes the noisy outcome, $g(.)$ denotes a monotone function, and w is the stochastic noise term. The interesting but also limiting facet of the Robbins-Monro algorithm is that the parameters, $\{a_n\}$, must constitute a sequence of step sizes that decrease over time, n. Unfortunately, this is a two-edged sword: While this is a *necessary* condition required to guarantee the scheme's convergence, it also leads to the simultaneous drawback that its renders the convergence to be slow[1].

These recursive algorithms were first introduced in the seminal paper by Robbins and Monro [13]. This work and a subsequent paper by Kiefer and Wolfowitz [3] are reckoned as fundamental to the family of stochastic approximation algorithms. Since then, an extensive body of literature on the SRF problem has emerged. For an exhaustive reference on stochastic approximation algorithms, we refer the reader to an excellent book by Spall [14] and the book by Kushner and Ying [4].

[1]It should be mentioned that a vast body of the literature has been focused on determining sequences for $\{a_n\}$ so as to enhance the convergence characteristics.

Drawbacks of stochastic approximation-based algorithms: The family of stochastic approximation-based algorithms are guaranteed to converge with probability 1 under some mild conditions. Indeed, it is worth mentioning that the rationale for algorithms that follow this paradigm is that the process of taking small step sizes creates an averaging effect on the noisy observations, and thus guides the optimization process in the right direction. However, this "small step" phenomenon is precisely why they suffer from a low convergence speed.

Mitigating the effects of small step sizes: Recently, Waeber and his colleagues [17], [16] have proposed a novel approach to solve the SRF problem which does not involve the theory of stochastic approximation, and more precisely the philosophy of "the step update". Their idea is based on a stochastic version of a solution to the *bisection search* which permits a more efficient exploration of the search space. Waeber and his colleagues showed through a rigorous theoretical endeavor and through experimental verification that their algorithm achieved a high rate of convergence and that it outperformed small step-update based algorithms [13], [4]. We applaud the work of Waeber and his colleagues in that they have ventured to propose a significantly different paradigm, implying that their schemes represent a quantum enhancement to the field. This is precisely the arena where this paper operates.

To motivate our paper, we mention that even though the works of Waeber and his colleagues are very encouraging, they resorted to strong assumptions that rather limited the strength of their schemes. More specifically, Waeber and his colleagues [17], [16] resorted to an unrealistic and very strong assumption that the probability of observing an incorrect sign in the outcomes $Y_n(x_n)$ is *known*. Clearly, this assumption is invalid in the vast majority of real-life scenarios.

As opposed to the original stochastic approximation paradigm, where the point investigated at any time instant is in the proximity of the point investigated at the previous time instant, we attempt to mitigate the effect of small step sizes by recognizing that $g()$ contains information using which large sections of the search space can be eliminated. Indeed, this information is unutilized in the paradigm that invokes stochastic approximation schemes. In this article, we propose to resolve this by using the theory of Learning Automata (LA), and more precisely, the Continuous Point Location with Adaptive d-ary Search (CPL-AdS) to intelligently prune the space by investigating d disjoint regions at each iterations. The CPL-AdS will be discussed, in fair detail, presently.

In contrast to the work done by the authors of [17], [16], we operate under the truly realistic assumptions that the probability of observing an incorrect sign in the outcomes $Y_n(x_n)$ is totally *unknown*. Thus, while the rationale of our scheme is similar to the philosophy of [17], [16], the one primary difference[2] is that we operate at epochs after which we can eliminate entire regions from the search space. More specifically, our solution recursively shrinks the search space by, at least, a factor of $\frac{2d}{3}$ at each epoch, where $d \geq 2$ is a user-defined parameter of the algorithm. This enhances the convergence significantly. Conceptually, this is achieved through a subtle re-formulation of the SRF problem in terms of a continuous-space generalization of the Stochastic Point Location (SPL) problem originally proposed by Oommen in [8]. For the rest of this paper, we will refer to our scheme as the Stochastic Root Finding with Adaptive d-ary Search (SRF-AdS).

Top-level explanation of our scheme: As we know, the goal of a SRF algorithm is to locate a point x^* such that g(x^*) = 0 for a given function g that can only be observed with noise. Our scheme queries the function g at a point x, and obtains a noisy measurement $Y(x) = g(x) + w(x)$, where w is an additive noise term. We will show later how the sign of $Y(x)$ holds noisy information about whether the root lies to the left or right of x^* – which constitutes the basis of our algorithm. If one considers the parameter that is sought for to be a "point" on the line, we can model this problem using the so-called SPL problem[3] alluded to above. In this paper, we will show how a subtle formulation of the SRF problem can be achieved in a manner by which we can obtain "signals" that point towards the correct direction of the optimal parameter along any dimension, and that this occurs with a probability greater than 0.5. This, consequently, leads to the proposed solution to the SRF.

II. Legacy SPL Solutions

To place our work in the right perspective, we briefly review[4] the state of the art of the SPL problem, whose formulation and solution is central to our approach. The SPL problem, in its most elementary formulation, assumes that there is a Learning Mechanism (LM) whose task is to determine the optimal value of some variable (or parameter), x. We assume that there is an optimal choice for x – an unknown value, say $x^* \in [0, 1)$. The SPL involves inferring the value x^*. Although the mechanism does not know the value of x^*, it was assumed that it has responses from an intelligent "Environment" (synonymously, referred to as the "Oracle"), Ξ, that is capable of informing it whether any value of x 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 x when it should be decreased, and *vice versa*. However, to render the problem tangible, in [8] 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 $x < x^*$, the Environment correctly suggests that we increase x with probability p. It simultaneously could have incorrectly recommended that we decrease x with probability $(1 - p)$. The converse is true for $x \geq x^*$.

[2]This must be contrasted with the works of [17], [16] in which the authors opted to update a *distribution* that reflects the certainty about the position of x^* in the line and showed that the mass of the distribution will converge to 1 in the neighborhood of x^* as time goes to infinity.

[3]The extension of the SPL to stochastic optimization problems was earlier alluded to in [8], [12], where the respective authors merely assumed the existence of an indicator as to the (approximate) value of the criterion function for any specified value of the parameter. However, no concrete strategy was specified as to how to model the response from the environment using an "Oracle".

[4]This review can be abridged or even deleted if requested by the Referees.

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

- Oommen [8] 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 was 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 ultimately implied a more accurate convergence to x^*. 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.

- An novel alternate *parallel* strategy that combined LA and pruning was used in [11] to solve the SPL. By utilizing the response from the environment, the authors of [11] 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[6].

- In a subsequent work [12], Oommen *et al.* introduced the Continuous Point Location with Adaptive d-ARY Search (CPL-AdS) which was a generalization of the work in [11]. In CPL-AdS, the given search interval was sub-divided into d partitions representing d disjoint subintervals, where $d > 3$. In each interval, initially, the midpoint of the given interval was considered to be the estimate of the unknown x^*. Each of the d partitions of the interval was independently explored using an ϵ-optimal two-action LA, where the two actions were those of selecting a point from the left or right half of the partition under consideration. Thereafter, the scheme proposed in [12] eliminated at least one of the subintervals from being searched further, and recursively searched the remaining pruned contiguous interval until the search interval was at least as small as the required resolution of estimation. Again, this elimination process essentially utilized the ϵ-optimality property of the underlying LA and the monotonicity of the intervals to guarantee the convergence. By virtue of this property, at each epoch consisting of a certain number, N_∞, of iterations, the algorithm could "$(1-\epsilon)$-confidently" discard regions of the search space.

- The authors of [2] proposed a rather straightforward modification of the latter CPL-AdS so as to also track changes in x^*. Indeed, to achieve the latter, the authors of [2] proposed to perform an *additional* parallel d-ARY search at each epoch on the original search interval. The limitation of this work is that the strategy proposed in [2] can only track x^* under certain conditions relative to the frequency of change in x^* and the length of an epoch. However, more importantly, the interesting facet of the solution presented in [12] 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.

- Recently Yazidi *et al.* [18] proposed a *hierarchical* searching scheme for solving the SPL problem. The solution involves 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. With regard to its advantages, this solution is an order of magnitude faster than the classical SPL solution [8]. The marginal drawback, however, is that it works under the premise that p is a constant whose value is larger than the golden ratio conjugate. Generalizing the solution proposed in [18] to the SRF is open. Indeed, it is far from trivial.

III. Problem Statement of Stochastic Root-Finding:

We shall first formalize the SRF problem, proceed to present the notation that we shall use and then present our solution.

Let g be a monotone function defined over the interval $\Delta = [\sigma, \gamma)$ such that there exists a unique point $x^* \in \Delta$ with $g(x^*) = 0$. The goal of our exercise is to locate the point x^*. The problem is non-trivial because the function g cannot be observed directly. Rather, we must glean information about g via a stochastic simulation phase where x is a control parameter of the simulator. For any $x \in \Delta$ the simulator produces (or rather, yields) random outcomes $Y(x) = g(x) + w \in R$, where w represents stochastic noise. Although the distributional form of w may be unknown, two common acceptable assumptions are that this distribution is symmetric and that $E(w) = 0$, .

Without loss of generality, we assume that g is monotonically decreasing[7] implying that $g(x) > 0$ for all $x < x^*$ and that $g(x) < 0$ for all $x > x^*$. This allows us to reformulate the problem by defining the function $r(x, x^*) = Prob(Y(x) \geq 0)$, as follows. First of all, it is easy to note that $r(x, x^*) > \frac{1}{2}$ for all $x < x^*$, and $r(x, x^*) < \frac{1}{2}$ for all $x > x^*$. Further, $r(x, x^*) = \frac{1}{2}$ for $x = x^*$. The reader will observe that unlike in the function $g(\cdot)$, we have specifically include x^* as an argument of $r(\cdot, \cdot)$ to emphasize that the response depends on both the point queried, x, and the location of the root, x^*. The LA-based algorithm for the SRF that we introduce in this paper uses only $Z(x) = sign(Y(x))$ when inferring the knowledge about g. In this case, the information exploited is simply whether x^* is to the left or right of x, and this "directional" information may be wrong with a certain probability. We shall soon argue that discarding information is counterproductive, because the magnitude of $Y(x)$ contains additional information about $g(x)$.

To aid in the formulation, we define the functions

[5] Some of the existing results about discretized automata are found in [1], [5], [7], [9], [10], [6], [15]. Indeed, the fastest reported LAs are the discretized pursuit, and discretized maximum likelihood and Bayesian estimator algorithms [1], [10], [6].

[6] The logic behind this is explained in the next item, when the authors generalized this scenario for the case when the number of partitions was $d > 3$.

[7] The case when it is monotonically increasing follows using the mirrored arguments and is thus easy to tackle based on the same approach that we present here.

$$p(x, x^*) := max(r(x, x^*), 1 - r(x, x^*)), \text{ and}$$

$$q(x, x^*) := 1 - p(x, x^*).$$

Clearly, p specifies the probability that the Oracle provides a correct answer. By considering the definitions of the functions r and g, it follows that:

$$p(x, x^*) > \tfrac{1}{2} \text{ for } x \neq x^*, \text{ and } p(x^*, x^*) = \tfrac{1}{2}.$$

The main problem with the solution of the SRF problem proposed by [17], [16] is that they assume that after sampling at x, the value of $p(x, x^*)$ is revealed. This is unrealistic, since, in practice, one is forced to estimate $p(x, x^*)$. The authors of [17], [16] have chosen to leave the realistic scenario when $p(x, x^*)$ is not revealed, as an avenue for for future research.

A. Notations and Definitions

In this section, we shall present the notations and definitions that we will use, and proceed to develop our LA-based solution. *From a cursory perspective, it appears as if the model and solution are identical to those of the CPL-AdS given in [12]. While the notation and formulation appear identical, the problems themselves and the consequent partitioning are quite distinct*. Indeed, the fundamental differences can be summarized as below:

- Unlike the classical SPL problem, in the SRF, the probability, p, of the "environment" suggesting an accurate response, is shown to be non-constant. The fact that p is not constant renders the analysis to be much more involved than in the cases analyzed in [12].
- It has to emphasized that the table that displays the partitioning and the rules for eliminating the sub-regions are completely distinct from those used in [12]. This, as clarified presently, is a consequence of the fact that, unlike in the CPL-AdS [12], p is not constant.
- In contrast to the classical SPL problem, the responses from the "environment" on whether to move *Right* or *Left* are asymmetric. This is absolutely not the case in the SPL.

Notation:

Let $\Delta = [\sigma, \gamma)$ s.t. $\sigma \leq x^\star < \gamma$ be the current search interval containing x^* whose left and right (smaller and greater) boundaries on the real line are σ and γ respectively. We partition Δ into d equi-sized[8] disjoint partitions Δ^j, $j \in \{1, 2, \ldots d\}$, such that, $\Delta^j = [\sigma^j, \gamma^j)$. To formally describe the relative locations of the intervals we define an interval relational operator $\prec$ such that, $\Delta^j \prec \Delta^k$ iff $\gamma^j < \sigma^k$. Since points on the real interval are monotonically increasing, we have, $\Delta^1 \prec \Delta^2 \ldots \prec \Delta^d$. For every partition Δ^j , we define L^j and R^j as its *Left* half and *Right* half respectively as:

$$L^j = \{x \,|\, \sigma^j \leq x < mid(\Delta^j)\}, \text{ and}$$
$$R^j = \{x \,|\, mid(\Delta^j) \leq x < \gamma^j\},$$

where $mid(\Delta^j)$ is the mid-point of Δ^j. A point $x \in L^j$ will be denoted by x_L^j, and a point $x \in R^j$ by x_R^j.

To relate the various intervals to x^*, we introduce the following relational operators.

$x^\star \circledcirc \Delta^j$ iff$x^\star < \sigma^j$.i.e., $x^\star$ is to the left of the interval Δ^j.
$x^\star \circledcirc \Delta^j$ iff$x^\star > \gamma^j$.i.e., $x^\star$ is to the right of the interval Δ^j.
$x^\star \circledcirc \Delta^j$ iff$\sigma^j \leq x^\star < \gamma^j$.i.e., $x^\star$ is contained in the interval Δ^j.

These operators can trivially be shown to satisfy the usual laws of transitivity.

B. Construction of the Learning Automata

In the SRF–AdS strategy, with each partition Δ^j we associate a 2-action L_{RI} automaton $\mathcal{A}^j$, $(\Sigma^j, \Pi^j, \Gamma^j, \Upsilon^j, \Omega^j)$ where, Σ^j is the set of actions, Π^j is the set of action probabilities, Γ^j is the set of feedback inputs from the Environment, Υ^j is the set of action probability updating rules, and Ω^j is the set of possible decision outputs of the automata at the end of each epoch. The Environment, E , is characterized by the probability of a correct response $p(x, x^*)$ which we shall later, analytically, map to the penalty probabilities, c_k^j, for the two actions of the automaton, $\mathcal{A}^j$. The overall search strategy SRF–AdS, in addition uses a decision table[9] Λ to prune the search interval by comparing the output decisions $\{\Omega^j\}$ for the d partitions. Thus $\mathcal{A}^j$, $j \in \{1, \ldots d\}$, together with E and Λ completely define the SRF–AdS strategy.

1) *The set of actions of the automaton*: (Σ^j)
   The two actions of the automaton are α_k^j, for $k \in \{0, 1\}$, where, α_0^j corresponds to selecting the *Left* half, L^j, of the partition Δ^j, and α_1^j corresponds to selecting the *Right* half, R^j.
2) *The action probabilities*: (Π^j)
   $P_k^j(n)$ represent the probabilities of selecting the action α_k^j, for $k \in \{0, 1\}$, at step n. Initially, $P_k^j(0) = 0.5$, for $k = 0, 1$.
3) *The feedback inputs from the Environment to each automaton*: (Γ^j)
   It is important to recognize a subtle, but crucial point in the construction of the learning automata in SRF–AdS. From the automaton's point of view, the two actions are those of selecting either the left or the right half of its partition. However, from the Environment's point of view, the automaton presents a current estimate x for the true value of x^*, and it gives a feedback based on the relative position (or direction) of x with respect to x^*. Thus, there is a need to map the intervals to a point value, and the feedback on the point value to the feedback on the choice of the intervals.
   Let the automaton select either the *Left* or *Right* half of the partition, and then pick a point randomly (using a continuous uniform probability distribution) from this sub-interval which is presented as the current estimate for $x^\star$. Then, the possible feedback values for $\beta(n)$ at step n are defined by the conditional probabilities:

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

[9] This table is also referred to as the "Pruning" Table.

$$\begin{aligned} Pr[\beta(n)=0 \,|\, x_L^j \in L^j \text{ and } x_L^j \geq x^\star] &= p(x_L^j, x^\star) \\ Pr[\beta(n)=0 \,|\, x_L^j \in L^j \text{ and } x_L^j < x^\star] &= q(x_L^j, x^\star) \\ Pr[\beta(n)=0 \,|\, x_R^j \in R^j \text{ and } x_R^j < x^\star] &= p(x_R^j, x^\star) \\ Pr[\beta(n)=0 \,|\, x_R^j \in R^j \text{ and } x_R^j \geq x^\star] &= q(x_R^j, x^\star) \end{aligned} \tag{1}$$

Note that, the condition $x_L^j \in L^j$ indicates that the action α_0^j was selected, and the condition $x_R^j \in R^j$ indicates the other action, α_1^j, was selected. The reader will also observe that we have tried to be consistent with the existing literature in which the response $\beta = 0$ is treated as a "Reward", and the response $\beta = 1$ is treated as a "Penalty".

- The action α_0^j (i.e., the one that corresponds to selecting the *Left* half, L^j, of the partition Δ^j) is rewarded whenever the LA chooses a point x_L^j in the left-half of the region, and Environment advices it to go to the left, meaning that $Y(x_L^j) < 0$.
- The action α_1^j (i.e., the one that corresponds to selecting the *Right* half, L^j, of the partition Δ^j) is rewarded whenever the LA chooses a point x_R^j in the right-half of the region, and the Environment advices it to go to the right, meaning that $Y(x_R^j) \geq 0$.

4) *The action probability updating rules*: (Υ^j)
First of all, since we are using the L_{RI} scheme, we ignore all the penalty responses. Upon reward, we obey the following updating rule:
If α_k^j for $k \in \{0, 1\}$was rewarded then,

$$\begin{aligned} P_{1-k}^j(n+1) &\leftarrow \theta \times P_{1-k}^j(n) \\ P_k^j(n+1) &\leftarrow 1 - \theta \times P_{1-k}^j(n), \end{aligned}$$

where $0 \ll \theta < 1$ is the L_{RI} reward parameter.

5) *The decision outputs at each epoch*: (Ω^j)
From the action probabilities we infer the decision Ω^j of the L_{RI} automaton, $\mathcal{A}^j$, after a fixed number N_∞, of iterations. This is referred to as an "Epoch". Typically, N_∞ is chosen so as to ensure (with a very high probability) that the automaton will have converged. Ω^j indicates that the automaton has inferred whether x^* is to the *Left*, *Right* or *Inside* the partition. The set of values that Ω^j can take and the preconditions are:

$$\Omega^j = \begin{cases} Left & \text{If} \quad P_0^j(N_\infty) \geq 1-\epsilon, \\ Right & \text{If} \quad P_1^j(N_\infty) \geq 1-\epsilon, \\ Inside & \text{Otherwise.} \end{cases}$$

6) *The decision table for pruning the search space*: (Λ)
Since the actions chosen by each LA can lead to one of three decisions, namely $Left, Inside$, or $Right$, the set of possible values in the decision table has cardinality 3^d, where d is the number of partitions. Once the individual automata for the d partitions have made a decision regarding where they reckon x^* to be, the SRF–AdS reduces the size of the search interval by eliminating at least one of these partitions. The new pruned search interval, Δ^{new}, for the subsequent learning phase (epoch) is generated according to the pruning decision table, Λ, for the specific value of d, and is created based on the following rules:

a) The table has $d+1$ columns. In each row, the entry in the i^{th} column is the decision inferred from the specific LA, namely its decision whether x^* is $Inside$, to the $Left$ of, or to the $Right$ of the current interval.
b) In each row, the entry in the $(d+1)^{th}$ column is the decision about what the pruned interval should be. This decision is based on the collective decisions of all the LA, with the understanding that each of them operates in an ϵ-optimal manner.
c) A sequence of LA decisions will be termed $Inconsistent$ if:
   i) *Any* LA, A^i, decides that x^* is to its $Right$, but any other LA, A^j, with $j < i$ decides that $x^\star$ is to its $Left$, and *vice versa*.
   ii) *Any* LA, A^i, decides that x^* is to its $Left$, but any other LA, A^j, with $j > i$ decides that $x^\star$ is $Inside$, its interval.
   iii) *Any* LA, A^i, decides that x^* is to its $Right$, but any other LA, A^j, with $j < i$ decides that $x^\star$ is $Inside$, its interval.
   iv) *More than one* LA decide that x^* is $Inside$ its interval.
d) No row which represents a set of $Inconsistent$ decisions is included in the Pruning Table, Λ.
e) The pruned entry for the row with decisions $\{Left, Left \ldots Left\}$ is $LeftHalf(\Delta^1)$.
f) The pruned entry for the row with decisions $\{Right, Right \ldots Right\}$ is Δ^d.
g) If two consecutive LA A^j and A^{j+1} decide that $x^\star$ is to the $Right$ and $Left$ of their corresponding intervals respectively, the pruned interval is $\Delta^j \cup LeftHalf(\Delta^{j+1})$.
h) If any LA A^j converges to $Inside$, the pruned interval is $LeftHalf(\Delta^{j+1})$.

This table, Λ, is shown in Table I for $d = 2$, in Table II for $d = 3$, and in Table III for $d = 4$.

TABLE I. THE DECISION TABLE, (Λ), TO PRUNE THE SEARCH SPACE OF SRF–AdS FOR $d = 2$ BASED ON THE AUTOMATA OUTPUTS Ω^j. OBSERVE THAT THE TABLE HAS ONLY 5 *consistent* ROWS.

Ω^1	Ω^2	**New Sub-interval** Δ^{new}
$Left$	$Left$	$LeftHalf(\Delta^1)$
$Inside$	$Left$	$LeftHalf(\Delta^1)$
$Right$	$Left$	$\Delta^1 \cup LeftHalf(\Delta^2)$
$Right$	$Inside$	$LeftHalf(\Delta^2)$
$Right$	$Right$	Δ^2

The table indeed "prunes" the size of the interval, because many of the combinations that are potentially possible are $Inconsistent$, and occur with probability zero if we use an ϵ-optimal scheme. This pruned table will contain only $O(d)$ rows out of the 3^d possible rows that could occur. Thus, Table I for $d = 2$ contains only 5 out of the possible 9 combinations, Table II for $d = 3$ contains only 7

TABLE II. THE DECISION TABLE, (Λ), TO PRUNE THE SEARCH SPACE OF SRF–AdS FOR $d = 3$ BASED ON THE AUTOMATA OUTPUTS Ω^j. OBSERVE THAT THE TABLE HAS ONLY 7 *consistent* ROWS.

Ω^1	Ω^2	Ω^3	**New Sub-interval** Δ^{new}
$Left$	$Left$	$Left$	$LeftHalf(\Delta^1)$
$Inside$	$Left$	$Left$	$LeftHalf(\Delta^1)$
$Right$	$Left$	$Left$	$\Delta^1 \cup LeftHalf(\Delta^2)$
$Right$	$Inside$	$Left$	$LeftHalf(\Delta^2)$
$Right$	$Right$	$Left$	$\Delta^2 \cup LeftHalf(\Delta^3)$
$Right$	$Right$	$Inside$	$LeftHalf(\Delta^3)$
$Right$	$Right$	$Right$	Δ^3

TABLE III. THE DECISION TABLE, (Λ), TO PRUNE THE SEARCH SPACE OF SRF–AdS FOR $d = 4$ BASED ON THE AUTOMATA OUTPUTS Ω^j. OBSERVE THAT THE TABLE HAS ONLY 9 *consistent* ROWS.

Ω^1	Ω^2	Ω^3	Ω^4	**New Sub-interval** Δ^{new}
$Left$	$Left$	$Left$	$Left$	$LeftHalf(\Delta^1)$
$Inside$	$Left$	$Left$	$Left$	$LeftHalf(\Delta^1)$
$Right$	$Left$	$Left$	$Left$	$\Delta^1 \cup LeftHalf(\Delta^2)$
$Right$	$Inside$	$Left$	$Left$	$LeftHalf(\Delta^2)$
$Right$	$Right$	$Left$	$Left$	$\Delta^2 \cup LeftHalf(\Delta^3)$
$Right$	$Right$	$Inside$	$Left$	$LeftHalf(\Delta^3)$
$Right$	$Right$	$Right$	$Left$	$\Delta^3 \cup LeftHalf(\Delta^4)$
$Right$	$Right$	$Right$	$Inside$	$LeftHalf(\Delta^4)$
$Right$	$Right$	$Right$	$Right$	Δ^4

out of the possible 27 combinations, and Table III for $d = 4$ contains only 9 out of the possible 81 combinations. Similarly, for the other values of d, the decision table for the subset of the rows that can result from the convergence of the L_{RI} automata can be easily written down, and in each case, the pruning rule of the interval can also be easily determined, and will contain $O(d)$ rows - which is much less than 3^d rows.

C. Output Vector

In this section, we will define $2d + 1$ output vectors for the d automata $\mathcal{A}^j, j \in \{1, 2, \ldots d\}$ which are consistent with the decision table created using the rules specified in Section III-B. Theorem 3 will show that a decision table constructed using these $2d + 1$ output vectors is complete.

To aid in the analysis and explanation, we define the following output vector: $\vec{\Omega}'_i$ for $1 \leq i \leq d$ as:

$$\vec{\Omega}'_i = [\underbrace{Right, Right, .., Right}_{i-1 \text{ first components}}, Inside, \overbrace{Left, Left, ..., Left}^{\text{components number } i+1 \text{ to } d}]$$

In addition, $\vec{\Omega}_i$ is defined for $1 \leq i \leq d + 1$ as:

- $\vec{\Omega}_1 = [Left, Left, .., Left]$
- $\vec{\Omega}_i = [\underbrace{Right, Right, .., Right}_{i-1 \text{ first components}}, \overbrace{Left, Left, ..., Left}^{\text{components number } i \text{ to } d}]$, for $2 \leq i \leq d$
- $\vec{\Omega}_{d+1} = [Right, Right, .., Right]$.

IV. CONVERGENCE PROOF

Lemma 1 and Theorem 1 essentially use the ϵ-optimality property of L_{RI} automata to prove that they produce, w. p. 1, the correct decision output for each partition. Theorem 2 proves that the decision table is complete by considering all possible consistent output vectors and all the possible positions of x^* in Δ.

Theorem 3 is the basis of the decision table. Given an output vector, we use a reasoning based on the principle of elimination to determine the possible relative position of x^* within the d partitions that could have resulted in the considered output vector. Theorem 3 establishes that after elimination of one or more partitions, the remaining interval still contains $x^\star$w. p. 1., thereby ensuring convergence. The reader should remember that all these claims are probabilistic results, and that the probability of convergence to the optimal partition can be as close to unity as we want, provided that we choose the parameters for the L_{RI} automata appropriately.

The proof of the theoretical results are quite involved and lengthy and so are omitted here due to the space limitations.. The complete proofs are found in the unabridged version of this paper [19].

We first state a fundamental result for L_{RI} learning schemes which we will repeatedly allude to in the rest of the paper.

Lemma 1: An L_{RI} learning scheme with parameter $0 \ll \theta < 1$ is ϵ-optimal, whenever an optimal action exists. In other words, if α^j_k is the optimal action, $\lim_{\theta\to 1} \lim_{N\to\infty} P^j_k(N) \to 1$.

Theorem 1: Given the L_{RI} scheme with a parameter θ which is arbitrarily close to unity, the following is true:

If $(x^\star \circledcirc \Delta^j)$, then $Pr(\Omega^j = Left) \to 1$.
If $(x^\star \circledcirc \Delta^j)$, then $Pr(\Omega^j = Right) \to 1$.
If $(x^\star \circledcirc$ RightHalf(Δ^j)), then $Pr(\Omega^j = Right) \to 1$.
If $(x^\star \circledcirc$ LeftHalf(Δ^j)), then
$Pr(\Omega^j = \{Left,\ Inside\ or\ Right\}) \to 1$.

Theorem 2: The decision table constructed by the $2d + 1$ output vectors defined by $\{\vec{\Omega}_i | 1 \leq i \leq d{+}1\} \bigcup \{\vec{\Omega}'_i | 1 \leq i \leq d\}$ is complete.

Theorem 3: If the algorithm uses the same L_{RI} scheme at all levels of the recursion with a parameter θ that is arbitrarily close to unity, and if N_∞ is sufficiently large, the unknown x^* is always contained (w. p. 1) in the new search-interval, Δ^{new} resulting from the application of the decision rules specified in Section III-B.

Theorem 4: SRF–AdS shrinks the search space by, at least, a factor of $\frac{2d}{3}$ at each epoch, where $d \geq 2$ is a user-defined parameter of the algorithm.

V. EXPERIMENTAL RESULTS

The stochastic root finding mechanism, SRF–AdS, described in the earlier sections, 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 various conditions, simulation experiments were conducted for various values of θ, the reward factor of the L_{RI} automata, and for two different noisy functions (linear and exponential) and for different values d, the number of partition made at each epoch. In all the experiments that we report, it was assumed that $x^\star \in [-5, 5)$, which constituted the original search interval, and this was used as the starting "point" of the scheme. Each epoch consisted of 250 iterations (N_∞) of the d L_{RI} automata. At the end of each epoch the decision table was consulted to prune the current search interval, and the algorithm was recursively invoked. The recursion was terminated when the width of the interval was less than twice the desired accuracy.

The results of our experiments are truly conclusive and confirm the power of the SRF–AdS scheme. Although several experiments were conducted using various $x^\star$ and parameter values, we report for brevity sake, only the results for two functions, the first being linear and the second, exponential. An ensemble of several independent replications with different random number streams were performed to minimize the variance of the reported results. The reported results are averaged over the ensemble of replications.

The most important issue that has to be emphasized is that the scheme *does, indeed, converge accurately*. This is not something that should be taken for granted, because, unlike the traditional small-step approaches surveyed earlier, we do not calculte the estimate for the root at the next iteration to be in the proximity of the estimate at the current iteration. Rather, we have chosen to take the daring step of discarding large segments of the search space, which could potentially be catastrophic. But, as the theorems confirm, the probability of discarding the correct sub-interval is arbitrarily small, and thus, as the epochs proceed, the interval that contains the root becomes progressively, geometrically, smaller. The experimental results reported below confirm this even when the variance of the noise is significant.

To report our results, we considered the following two functions:

$$g_1(x) = -9x + 3, g_2(x) = exp(-5x) - 4.$$

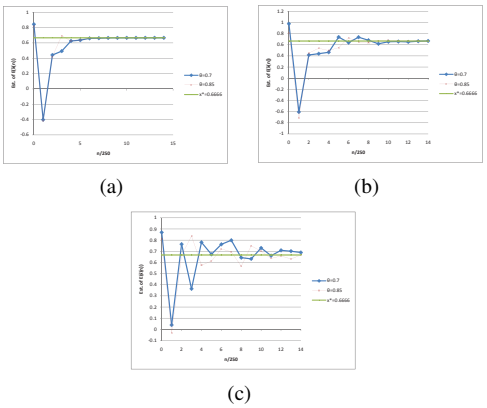
Note that g_1 admits a root at $x_1^* = 2/3 \approx 0.666$ and g_2 has its root at $x_2^* = -Ln(4)/5 \approx -0.27725887$. We chose $\theta = 0.8$, the initial search interval was $[-5, 5)$, with $N_\infty = 250$. The noise was normally distributed characterized by $N(0, \sigma)$. The spectrum of experiments was done by varying θ and the standard deviation, σ, where a larger value of σ implied a higher level of noise.

A. Experiments for Tertiary Search

In the first set of experiments, we chose a tertiary pruning scheme by fixing d to 3 to solve g_1 and g_2.

1) Linear Function: To demonstrate the power of the SRF scheme, we present the variation of $\hat{E}[\hat{x}(n)]$ with time n for the linear function g_1 (whose root is $x_1^* = 0.666$) for different types of noise. The variation of the solution is shown as a function of time n measured in epochs of size 250 units. The results that we plot are displayed in Figures 1(a), 1(b) and 1(c), where the standard deviations of the noise are $\sigma = 0.2$, $\sigma = 0.7$ and $\sigma = 1.0$ respectively.

The reader must observe, first of all, that the algorithms converged to the true root *in every single case,* and that in every epoch, the *search space was decreased significantly*. One must also observe that as we increased the noise steadily from 0.2 to 1.0, the convergence speed decreased – which was as we expected. Finally, in addition, we obtained a slight increase in the convergence speed when the noise parameter was fixed but as the parameter θ was increased. This phenomenon can also be seen from Figures 1(a), Figure 1(b) and Figure 1(c).



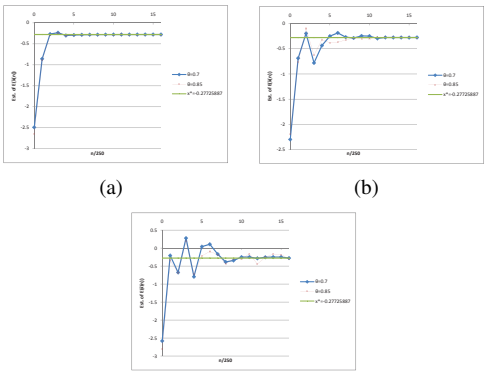
(a) (b)

(c)

Fig. 1. This figure depicts the variation of $\hat{E}[\hat{x}(n)]$ with time n, when the L_{RI} reward factor varies from $\theta = 0.7$ to $\theta = 0.85$. Here $x_1^* = 0.666$. The time n is shown in epochs of size 250 units. The standard deviation of the noise increases as (a) $\sigma = 0.2$, (b) $\sigma = 0.7$ and (c) $\sigma = 1.0$.

2) Non-Linear Function: In the same vein as the previous experiment, we examine here the variation of $\hat{E}[\hat{x}(n)]$ with time n for the non-linear function g_2 when the level of noise was steadily increased. The value of the standard deviations and the epoch lengths were the same as in the earlier case.

The variations are plotted n Figures 2(a), 2(b) and 2(c) for the scenarios when the standard deviations of the noise were respectively $\sigma = 0.2$, $\sigma = 0.7$ and $\sigma = 1.0$. Again, we observe that as expected, as we increased the noise steadily from 0.2, to 1.0, the convergence speed decreased. In addition, we observe that there was an increased speed in the convergence (for a specific noise level) as we increased θ. This can be seen from Figures 2(a), 2(b) and 2(c). But in every case, one should note that SRF–AdS converged to the true but unknown root.



(a) (b)

(c)

Fig. 2. This figure depicts the variation of $\hat{E}[\hat{x}(n)]$ with time n, when the L_{RI} reward factor varies from $\theta = 0.7$ to $\theta = 0.85$. Here $x_2^* = -Ln(4)/5 \approx -0.27725887$. The time n is shown in epochs of size 250 units. The standard deviation of the noise increases as (a) $\sigma = 0.2$, (b) $\sigma = 0.7$ and (c) $\sigma = 1.0$.

VI. Conclusion

In this paper we have considered the problem of solving the Stochastic Root Finding (SRF) problem, which is the most fundamental problem encountered in the field of stochastic optimization. The problem involves the the task of locating an unknown point x^* for which $g(x^*) = 0$ for a given function g that can only be observed in the presence of noise [13]. The traditional stochastic approximation solutions, reported for more than five decades, operate in a conservative manner by means of so-called "small-step" processes that incrementally explore the search space. Using this paradigm, the point investigated at any time instant is in the proximity of the point investigated at the previous time instant, rendering the convergence towards the optimal point, x^*, to be sluggish. This paper provides a pioneering and novel scheme to discover and utilize information using which large sections of the search space can be eliminated. Our solution recursively shrinks the search space by, at least, a factor of $\frac{2d}{3}$ at each epoch, where $d \geq 2$ is a user-defined parameter of the algorithm. This enhances the convergence significantly.

The method that we have proposed is akin to the solution to the Stochastic Point Location (SPL) problem originally proposed by Oommen in [8], and in particular to the Continuous Point Location with Adaptive d-ary Search (CPL–AdS), originally presented in [12]. However, since the latter is not applicable in our particular domain because of the inherent asymmetry of the SRF problem, it requires a completely new pruning strategy. Indeed, in contrast to the search on the line problem , our theoretical results are much more involved because the probability that the Environment correctly informs the LA about the location of the root is no more assumed to be a fixed quantity, p. In fact, in our case, the latter quantity depends on the sampled point, x. To the best of our knowledge, this paper presents the first LA-based solution to the SRF problem.

Recently Yazidi *et al.* [18] proposed a hierarchical searching scheme for solving the SPL problem. The solution involves partitioning the line in a hierarchical tree-like manner, and moving to relatively distant points, as characterized by those along the path of the tree. The solution proposed in [18] is an order of magnitude faster than classical SPL solution [8]; however, it works under the premise that p is constant and larger than the golden ratio conjugate. Generalizing the latter solution to the SRF problem is currently being investigated, although it is far from trivial.

References

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

[2] D.-S. Huang and W. Jiang. A general cpl-ads methodology for fixing dynamic parameters in dual environments. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 42(5):1489 –1500, October 2012.

[3] J. Kiefer, J. Wolfowitz, et al. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[4] H. J. Kushner and G. Yin. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer, 2003.

[5] J. K. Lanctôt and B. J. Oommen. Discretized estimator learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-22(6):1473–1483, November/December 1992.

[6] B. Oommen and M. Agache. Continuous and discretized pursuit learning schemes: various algorithms and their comparison. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 31(3):277 –287, June 2001.

[7] B. J. Oommen. Absorbing and ergodic discretized two-action learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-16:282–293, March/April 1986.

[8] B. J. Oommen. Stochastic searching on the line and its applications to parameter learning in nonlinear optimization. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-27B:733–739, 1997.

[9] B. J. Oommen and E. Hansen. The asymptotic optimality of discretized linear reward-inaction learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-14(3), May/June 1986.

[10] B. J. Oommen and J. K. Lanctôt. Discretized pursuit learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-20(4):931–938, July/August 1990.

[11] B. J. Oommen and G. Raghunath. Automata learning and intelligent tertiary searching for stochastic point location. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-28B:947–954, 1998.

[12] B. J. Oommen, G. Raghunath, and B. Kuipers. Parameter learning from stochastic teachers and stochastic compulsive liars. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-36B:820–836, 2006.

[13] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[14] J. C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*, volume 65. John Wiley & Sons, 2005.

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

[16] R. Waeber, P. I. Frazier, and S. G. Henderson. A bayesian approach to stochastic root finding. In *Simulation Conference (WSC), Proceedings of the 2011 Winter*, pages 4033–4045. IEEE, 2011.

[17] R. Waeber, P. I. Frazier, and S. G. Henderson. Bisection search with noisy responses. *SIAM Journal on Control and Optimization*, 51(3):2261–2279, 2013.

[18] A. Yazidi, O. Granmo, B. John Oommen, and M. Goodwin. A novel strategy for solving the stochastic point location problem using a hierarchical searching scheme. *IEEE Transactions on Cybernetics*, 44(11):2202–2220, Nov 2014.

[19] A. Yazidi and B. J. Oommen. A novel technique for stochastic root-finding: Enhancing the search with adaptive d-ary search. *Unabridged journal version of this paper, 2015. To be submitted for publication.*