A Novel Technique for Stochastic Root-Finding: Enhancing the Search with Adaptive d-ary Search[*]

Anis Yazidi[†], and B. John Oommen[‡]

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 [15]. 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 [9]. Our scheme is based, in part, on the Continuous Point Location with Adaptive d-ary Search (CPL–AdS), originally presented in [13]. The solution to the CPL–AdS [13], 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* [9, 13], 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 [13]. The decision rules for pruning are also different from those encountered when p is constant [13].

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

[*]The second author is grateful for the partial support provided by NSERC, the Natural Sciences and Engineering Research Council of Canada.

[†]This author can be contacted at: Dept. of Computer Science, University College of Oslo and Akershus, Oslo. E-mail: anis.yazidi@hioa.no.

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

1 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 [15] 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:

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

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

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, $w(n)$ is the stochastic noise term at the time instant n. 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 [15]. 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 [17] and the book by Kushner and Ying [4].

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 [19, 20] 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 [4, 15]. 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.

[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.

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 [19, 20] 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 [19, 20], 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 [19, 20], 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 [9]. 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 $Y(x) = g(x) + w$, 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.

1.1 Contributions

The novel contributions of this paper are listed below:

- We present a solution to the SPL problem which bridges the gap between two unrelated fields in computer

[2]This must be contrasted with the works of [19, 20] 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 [9,13], 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".

science, namely those related to the SPL (a LA-based problem) and the Stochastic Root Finding problem, which is, in and of itself, a stochastic optimization problem.

- In contrast to the formulation of the classical SPL problem proposed by Oommen *et al*, in our setting, 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 [13]. Interestingly, the decision rules for pruning encountered here are also different than those seen in the case of a constant p [13].
- Also, in contrast to the classical SPL problem, the responses from the "environment" on whether to move *Right* or *Left* are asymmetric, which renders its generalization to the SRF problem non-trivial.
- We present a thorough theoretical analysis that demonstrates that our scheme is ϵ-optimal and that it converges to the true root of the SRF problem with a probability that is arbitrarily close to unity.
- As a by-product of our analysis, we also demonstrate how we can evaluate, by computing well-defined definite integrals, the penalty probabilities associated with a problem modeled in a LA-setting.

2 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 [9] 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 $x \geq x^*$.

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

- Oommen [9] 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

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

[5]Some of the existing results about discretized automata are found in [1,6,8,10,11,14,18]. Indeed, the fastest reported LAs are the discretized pursuit, and discretized maximum likelihood and Bayesian estimator algorithms [1,11,14].

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 [12] to solve the SPL. By utilizing the response from the environment, the authors of [12] 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 [13], Oommen *et al.* introduced the Continuous Point Location with Adaptive d-ARY Search (CPL-AdS) which was a generalization of the work in [12]. 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 [13] 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 [13] 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.* [21] 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 [9]. 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 [21] to the SRF is open. Indeed, it is far from trivial.

[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 $d > 3$.

3 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

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

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

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 [19, 20] 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 [19, 20] have chosen to leave the realistic scenario when $p(x, x^*)$ is not revealed, as an avenue for for future research.

3.1 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*

[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.

given in [13]. 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 [13].
- 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 [13]. This, as clarified presently, is a consequence of the fact that, unlike in the CPL-AdS [13], 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.
- Unlike in the SPL, we demonstrate how we can evaluate, by computing well-defined definite integrals, the penalty probabilities associated with the SRF problem modeled in a LA-setting. This too is completely different from what one would encounter in the SPL.
- As a consequence of all the above, the proof that our scheme is ϵ-optimal (i.e., that it converges to the true root of the SRF problem with a probability that is arbitrarily close to unity) is far more intricate and distinct.

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^j_L, and a point $x \in R^j$ by x^j_R.

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

$$\begin{array}{llll}
x^\star \olessthan \Delta^j & \text{iff} & x^\star < \sigma^j. & \text{i.e., } x^\star \text{ is to the left of the interval } \Delta^j. \\
x^\star \ogreaterthan \Delta^j & \text{iff} & x^\star > \gamma^j. & \text{i.e., } x^\star \text{ is to the right of the interval } \Delta^j. \\
x^\star \circledequal \Delta^j & \text{iff} & \sigma^j \leq x^\star < \gamma^j. & \text{i.e., } x^\star \text{ is contained in the interval } \Delta^j.
\end{array}$$

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

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

3.2 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 $\mathcal{A}^j$, $j \in \{1, \dots d\}$, 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:

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

   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".

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

- 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,

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

   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 1 for $d = 2$, in Table 2 for $d = 3$, and in Table 3 for $d = 4$.

Table 1: 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 1 for $d = 2$ contains only 5 out of the possible 9 combinations, Table 2 for $d = 3$ contains only 7 out of the possible 27 combinations, and Table 3 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.

Table 2: 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 3: 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

3.3 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 3.2. 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 $2 \leq i \leq d$
- $\vec{\Omega}_{d+1} = [Right, Right, .., Right]$.

4 Convergence Proof

We shall now prove the convergence results concerning SRF–AdS. 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.

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_k^j is the optimal action, $\lim_{\theta\to 1}\lim_{N\to\infty} P_k^j(N) \to 1$.*

The above result is well known [5,7,16]. By virtue of this result, we are guaranteed that for any L_{RI} scheme with the two actions $\{\alpha_0, \alpha_1\}$, if $\exists k \in \{0,1\}$ such that $c_k^j < c_{1-k}^j$, then the action α_k^j is optimal, and for this action $P_k^j(N) \to 1$ as $N \to \infty$ and $\theta \to 1$. □

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

$$\begin{array}{ll}
\textit{If } (x^\star \olessthan \Delta^j), & \textit{then } Pr(\Omega^j = Left) \to 1. \\
\textit{If } (x^\star \ogreaterthan \Delta^j), & \textit{then } Pr(\Omega^j = Right) \to 1. \\
\textit{If } (x^\star \circledequal \textit{RightHalf}(\Delta^j)), & \textit{then } Pr(\Omega^j = Right) \to 1. \\
\textit{If } (x^\star \circledequal \textit{LeftHalf}(\Delta^j)), & \textit{then } Pr(\Omega^j = \{Left,\ Inside \textit{ or } Right\}) \to 1.
\end{array}$$

Proof:

Let us define $S^+ = \{x | x \geq x^*\}$, and $S^- = \{x | x < x^*\}$. Then, for a given x^* and for a given interval Δ^j, there are 4 mutually exclusive and exhaustive cases that can occur depending on the position of x^* relative to Δ^j. These are:

Case 1: Whenever $(x^\star \olessthan \Delta^j)$
Case 2: Whenever $(x^\star \ogreaterthan \Delta^j)$
Case 3: Whenever $(x^\star \circledequal$ RightHalf$(\Delta^j))$
Case 4: Whenever $(x^\star \circledequal$ LeftHalf$(\Delta^j))$

To simplify the notation, since we are always dealing with the same root x^*, we shall simplify the notation and consistently use $p(x)$ to imply $p(x, x^*)$.

Table 4: The different cases of the expression of the penalty probability at the sampled point x

Penalty	position of x
$1 - p(x)$	$L_j \cap S^+$
$p(x)$	$L_j \cap S^-$
$1 - p(x)$	$R_j \cap S^-$
$p(x)$	$R_j \cap S^+$

In order to ease the understanding of the proof, we provide the expressions of the penalty probability of the LA for all different positions of the sampled point x in Table 4.

The essence of the reward and penalty philosophy can explained in simple terms. If we choose a point x from the right-half region (i.e $x \in R_j$) and x^* is truly to the right of x (i.e $x \in S^-$), then the Environment suggests that x^* is to the right of x (i.e $Y(x) \geq 0$) with probability $p(x)$. Thus the penalty in the case where $x \in R_j \cap S^-$, is $1 - p(x)$. Similarly, if we choose a point x from the left-half region (i.e $x \in L_j$) and x^* is truly to the left x (i.e $x \in S^+$), then the Environment suggests that x^* is to the left of x (i.e $Y(x) < 0$) with probability $p(x)$. Thus the penalty in the case where $x \in L_j \cap S^+$ is $1 - p(x)$.

Figure (1) depicts a typical case of the variation of $p(x)$. Note that $p(x) > 1/2$ for $x \neq x^*$ and $p(x^*) = 1/2$.

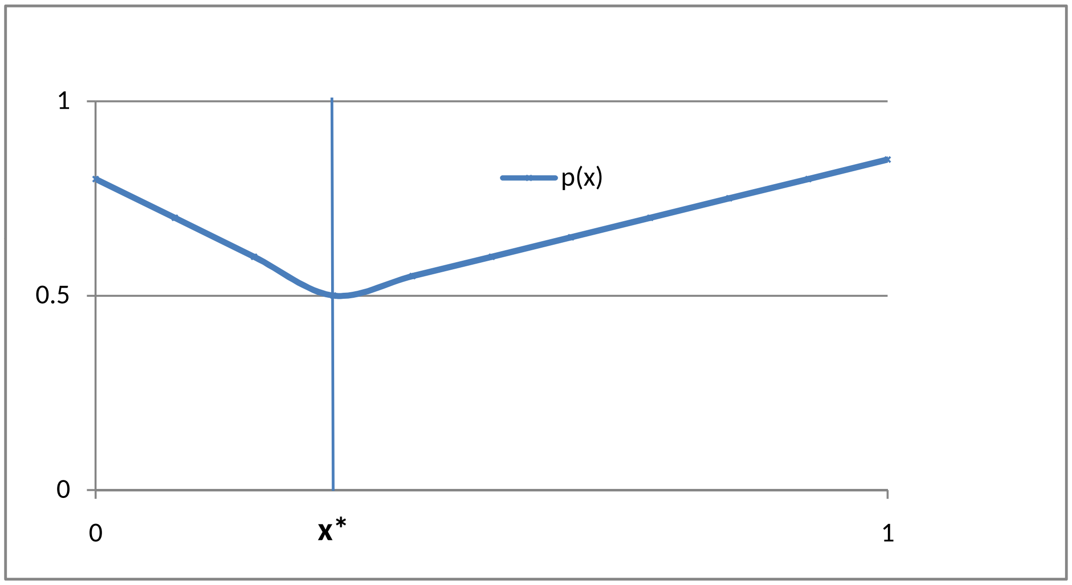


Figure 1: Variation of $p(x)$. Note that $p(x) \geq 1/2$.

We now proceed with the analysis of each of these cases individually.

Case 1: This is the case encountered when x^* is to the left of Δ^j, i.e., $x^* \otimes \Delta^j$ implying that $x^* < \sigma^j$.

By invoking the definition of the penalty probability c_0^j we have[10]:

$$\begin{aligned}
c_0^j &= Pr(\beta(n) = 1 \,|\, \text{sub-interval } L^j \text{ is chosen at step } n) \\
&= \int\limits_{L_j \cap S^+} (1-p(x))\,\mathrm{dP}(x) + \int\limits_{L_j \cap S^-} p(x)\,\mathrm{dP}(x) \\
&= \int\limits_{L_j} (1-p(x))\,\mathrm{dP}(x) \\
&= \int_{\sigma^j}^{mid(\Delta^j)} \frac{1-p(x)}{mid(\Delta^j)-\sigma^j}\,\mathrm{d}x.
\end{aligned}$$

Similarly, we derive c_1^j:

$$\begin{aligned}
c_1^j &= Pr(\beta(n) = 1 \,|\, \text{subinterval } R^j \text{ is chosen at step } n) \\
&= \int\limits_{R_j \cap S^-} (1-p(x))\,\mathrm{dP}(x) + \int\limits_{R_j \cap S^+} p(x)\,\mathrm{dP}(x) \\
&= \int\limits_{R_j} p(x)\,\mathrm{dP}(x) \\
&= \int_{mid(\Delta^j)}^{\gamma^j} \frac{p(x)}{\gamma^j - mid(\Delta^j)}\,\mathrm{d}x.
\end{aligned}$$

Our task is to now see how c_1^j compares with c_0^j. By virtue of the monotonicity property of the function g, we can guarantee that for $\forall\, x \in L^j$ (where the arbitrary point is denoted by x_L^j), and $\forall\, x \in R^j$ (where the arbitrary point is denoted by x_R^j), we have $p(x_R^j) > 1/2$ and $1 - p(x_L^j) < 1/2$.

Thus,

$$p(x_R^j) > 1 - p(x_L^j).$$

We now take into account the fact that $\gamma^j - mid(\Delta^j) = mid(\Delta^j) - \sigma^j$. This permits to deduce that:

$$\int_{\sigma^j}^{mid(\Delta^j)} \frac{1-p(x)}{mid(\Delta^j)-\sigma^j}\,\mathrm{d}x < \int_{mid(\Delta^j)}^{\gamma^j} \frac{p(x)}{\gamma^j - mid(\Delta^j)}\,\mathrm{d}x. \tag{2}$$

Therefore $c_0^j < c_1^j$. By now applying Lemma 1, we obtain that $Pr(\Omega^j = Left) \to 1$.

Case 2: This is the case encountered when $x^* \oslash \Delta^j$, i.e, $x^* > \gamma^j$, meaning that x^* is to the right of the interval Δ^j.

[10] In the expression below and in all the future similar expressions that involve an integral, dP denotes that the integral is taken with respect to the specific distribution of the sampled point.

The penalty probability c_0^j in this case can be written as:

$$\begin{aligned}
c_0^j &= Pr(\beta(n) = 1 \,|\, \text{subinterval } L^j \text{ is chosen at step } n) \\
&= \int\limits_{L_j \cap S^-} p(x)\,\mathrm{dP}(x) + \int\limits_{L_j \cap S^+} (1 - p(x))\,\mathrm{dP}(x) \\
&= \int\limits_{L_j} p(x)\,\mathrm{dP}(x) \\
&= \int_{\sigma^j}^{mid(\Delta^j)} \frac{p(x)}{mid(\Delta^j) - \sigma^j}\,\mathrm{d}x.
\end{aligned}$$

Similarly, we derive c_1^j:

$$\begin{aligned}
c_1^j &= Pr(\beta(n) = 1 \,|\, \text{sub-interval } R^j \text{ is chosen at step } n) \\
&= \int\limits_{R_j \cap S^-} (1 - p(x))\,\mathrm{dP}(x) + \int\limits_{R_j \cap S^+} p(x)\,\mathrm{dP}(x) \\
&= \int\limits_{R_j} 1 - p(x)\,\mathrm{dP}(x) \\
&= \int_{mid(\Delta^j)}^{\gamma^j} \frac{1 - p(x)}{\gamma^j - mid(\Delta^j)}\,\mathrm{d}x.
\end{aligned}$$

In this case, we can prove that $c_0^j < c_1^j$. Indeed, the monotonicity property of the function g guarantees that for $\forall\, x \in L^j$ (where the arbitrary point is denoted by x_L^j), and any point $x \in R^j$ (where the arbitrary point is denoted by x_R^j): $1 - p(x_R^j) < 1/2$ and $p(x_L^j) > 1/2$.

Thus,

$$1 - p(x_R^j) < p(x_L^j)$$

Again, taking into account the fact that $\gamma^j - mid(\Delta^j) = mid(\Delta^j) - \sigma^j$, we can deduce that:

$$\int_{mid(\Delta^j)}^{\gamma^j} \frac{1 - p(x)}{\gamma^j - mid(\Delta^j)}\,\mathrm{d}x < \int_{\sigma^j}^{mid(\Delta^j)} \frac{p(x)}{mid(\Delta^j) - \sigma^j}\,\mathrm{d}x.$$

implying that $c_1^j < c_0^j$. Again, by applying Lemma 1, we obtain that $Pr(\Omega^j = Right) \to 1$.

Case 3: We now consider the case when $x^* \ominus$ RightHalf(Δ^j). In other words, x^* is in R^j or equivalently $mid(\Delta^j) \leq x^* < \gamma^j$. As before, we derive the expressions of c_0^j and c_1^j below:

$$\begin{aligned}
c_0^j &= Pr(\beta(n) = 1 \,|\, \text{sub-interval } L^j \text{ is chosen at step } n) \\
&= \int\limits_{L_j \cap S^+} (1 - p(x))\, \mathrm{dP}(x) + \int\limits_{L_j \cap S^-} p(x)\, \mathrm{dP}(x) \\
&= 0 + \int_{\sigma^j}^{mid(\Delta^j)} p(x)\, \mathrm{dP}(x) \\
&= \int_{\sigma^j}^{mid(\Delta^j)} \frac{p(x)}{\sigma^j - mid(\Delta^j)}\, \mathrm{d}x.
\end{aligned}$$

Let us now consider c_1^j.

$$\begin{aligned}
c_1^j &= Pr(\beta(n) = 1 \,|\, \text{sub-interval } R^j \text{ is chosen at step } n) \\
&= \int\limits_{R_j \cap S^-} (1 - p(x))\, \mathrm{dP}(x) + \int\limits_{R_j \cap S^+} p(x)\, \mathrm{dP}(x) \\
&= \int_{x^*}^{\gamma^j} (1 - p(x))\, \mathrm{dP}(x) + \int_{mid(\Delta^j)}^{x^*} p(x)\, \mathrm{dP}(x) \\
&= \int_{x^*}^{\gamma^j} \frac{1 - p(x)}{\gamma^j - mid(\Delta^j)}\, \mathrm{d}x + \int_{mid(\Delta^j)}^{x^*} \frac{p(x)}{\gamma^j - mid(\Delta^j)}\, \mathrm{d}x.
\end{aligned}$$

We shall now consider how c_0^j compares with c_1^j. For the sake of the proof, we will re-write c_0^j:

$$\begin{aligned}
c_0^j &= \int_{\sigma^j}^{mid(\Delta^j)} \frac{p(x)}{\sigma^j - mid(\Delta^j)}\, \mathrm{d}x \\
&= \int_{\sigma^j}^{\sigma^j + (x^* - mid(\Delta^j))} \frac{p(x)}{\sigma^j - mid(\Delta^j)}\, \mathrm{d}x + \int_{\sigma^j + (x^* - mid(\Delta^j))}^{mid(\Delta^j)} \frac{p(x)}{\sigma^j - mid(\Delta^j)}\, \mathrm{d}x.
\end{aligned}$$

The reader should note that $\sigma^j + (x^* - mid(\Delta^j))$ belongs to $Left(\Delta^j)$ since $\sigma^j < \sigma^j + (x^* - mid(\Delta^j)) < mid(\Delta^j)$. Since we know that p is monotonically decreasing over S^-, we can assert that p attains its minimum at x^*, where $p(x^*) = 1/2$. Further, since p is monotonically decreasing over S^-, we see that for all x_1 and x_2 in S^- such that $x_1 < x_2$, we have:

$$p(x_2) < p(x_1).$$

In particular, $\forall\, x_R^j \in [mid(\Delta^j), x^*]$ for $x_L^j \in [\sigma^j, \sigma^j + (x^* - mid(\Delta^j)]$, we know that $x_L^j < x_R^j$ and thus,

$$p(x_R^j) < p(x_L^j).$$

Let m be a an upper bound for $p(x_L^j)$ over the interval $[\sigma^j, \sigma^j + (x^* - mid(\Delta^j)]$, and a lower bound for $p(x_R^j)$

over the interval $[mid(\Delta^j), x^*]$. In other terms, m satisfies:

$$p(x_L^j) > m > p(x_R^j). \tag{3}$$

To examine the first half of the inequality in Eq. (3), we integrate over the relevant interval to get:

$$\begin{aligned}
p(x_L^j) > m \quad &\Rightarrow \quad \int_{\sigma^j}^{\sigma^j+(x^*-mid(\Delta^j))} \frac{p(x)}{\sigma^j - mid(\Delta^j)}\,\mathrm{d}x > (\sigma^j + (x^* - mid(\Delta^j)) - \sigma^j)\frac{m}{\sigma^j - mid(\Delta^j)} \quad (4)\\
&\Rightarrow \quad \int_{\sigma^j}^{\sigma^j+(x^*-mid(\Delta^j))} \frac{p(x)}{\sigma^j - mid(\Delta^j)}\,\mathrm{d}x > (\sigma^j + (x^* - mid(\Delta^j)) - \sigma^j)\frac{m}{\gamma^j - mid(\Delta^j)} \quad (5)\\
&\Rightarrow \quad \int_{\sigma^j}^{\sigma^j+(x^*-mid(\Delta^j))} \frac{p(x)}{\sigma^j - mid(\Delta^j)}\,\mathrm{d}x > (x^* - mid(\Delta^j))\frac{m}{\gamma^j - mid(\Delta^j)}. \quad (6)
\end{aligned}$$

Note that, in the above, we applied the fact that $\sigma^j - mid(\Delta^j) = \gamma^j - mid(\Delta^j)$ since $mid(\Delta^j)$ is the midpoint of the interval.

In a similar manner, by considering the second half of the inequality in Eq. (3), (i.e., $m < p(x_L^j)$), we can prove that (to avoid being cumbersome, we have omitted the tedious algebraic manipulations involving the similar integrations):

$$(x^* - mid(\Delta^j))\frac{m}{\gamma^j - mid(\Delta^j)} > \int_{mid(\Delta^j)}^{x^*} \frac{p(x)}{\gamma^j - mid(\Delta^j)}\,\mathrm{d}x. \tag{7}$$

Combining both Eq. (6) and (7) we obtain that:

$$\int_{mid(\Delta^j)}^{x^*} \frac{p(x)}{\gamma^j - mid(\Delta^j)}\,\mathrm{d}x < \int_{\sigma^j}^{\sigma^j+(x^*-mid(\Delta^j))} \frac{p(x)}{\sigma^j - mid(\Delta^j)}\,\mathrm{d}x. \tag{8}$$

We know that $\forall\, x_1 \in [\sigma^j + (x^* - mid(\Delta^j)), mid(\Delta^j)]$ and $\forall\, x_2 \in [x^*, \gamma^j]$,

$p(x_1) > 1 - p(x_2)$

since $p(x_1) > 1/2$ and $1 - p(x_2) < 1/2$. Using the fact that the limits of the intervals, i.e., $[\sigma^j + (x^* - mid(\Delta^j)), mid(\Delta^j)]$ and $[x^*, \gamma^j]$ have the same length, one can see that:

$$\int_{x^*}^{\gamma^j} \frac{1 - p(x)}{\gamma^j - mid(\Delta^j)}\,\mathrm{d}x < \int_{\sigma^j+(x^*-mid(\Delta^j))}^{mid(\Delta^j)} \frac{p(x)}{\sigma^j - mid(\Delta^j)}\,\mathrm{d}x. \tag{9}$$

Note that $p(x_1) = 1 - p(x_2)$ if and only if $x_1 = x_2 = x^*$, which is a situation that can never occur if x_1 and x_2 are not in the same interval, implying that the above inequality is strict. Thus, using both the inequality given by Eq. (8) and inequality given by Eq. (9), and by summing up the left terms and the right terms together, we prove that $c_1^j < c_0^j$. By applying Lemma 1, we obtain the result that $Pr(\Omega^j = Right) \to 1$.

Case 4: This is the scenario that occurs when x^* is in L^j and when $\sigma^j \leq x^* < mid(\Delta^j)$, i.e., x^* is in $LeftHalf(\Delta^j)$. Computing c_0^j and c_1^j as before:

$$\begin{aligned}
c_0^j &= Pr(\beta(n) = 1 \,|\, \text{sub-interval } L^j \text{ is chosen at step } n) \\
&= \int\limits_{L_j \cap S^-} (1 - p(x))\, \mathrm{dP}(x) + \int\limits_{L_j \cap S^+} p(x)\, \mathrm{dP}(x) \\
&= \int_{x^*}^{mid(\Delta^j)} (1 - p(x))\, \mathrm{dP}(x) + \int_{\sigma^j}^{x^*} p(x)\, \mathrm{dP}(x) \\
&= \int_{x^*}^{mid(\Delta^j)} \frac{1 - p(x)}{mid(\Delta^j) - \sigma^j} \mathrm{d}x + \int_{\sigma^j}^{x^*} \frac{p(x)}{mid(\Delta^j) - \sigma^j}\, \mathrm{d}x.
\end{aligned}$$

$$\begin{aligned}
c_1^j &= Pr(\beta(n) = 1 \,|\, \text{sub-interval } R^j \text{ is chosen at step } n) \\
&= \int\limits_{R_j \cap S^+} (1 - p(x))\, \mathrm{dP}(x) + \int\limits_{R_j \cap S^-} p(x)\, \mathrm{dP}(x) \\
&= \int_{mid(\Delta^j)}^{\gamma^j} p(x)\, \mathrm{dP}(x) \\
&= \int_{mid(\Delta^j)}^{\gamma^j} \frac{p(x)}{\gamma^j - mid(\Delta^j)}\, \mathrm{d}x.
\end{aligned}$$

The formal proof from here is quite involved. Indeed, we will prove that depending on the relative position of x^* in the interval L^j, the outcome will lead to one of the three following cases:

$c_1^j < c_0^j$, or $c_1^j > c_0^j$ or $c_0^j = c_1^j$.

To explicitly clarify that c_0^j is a function of x^*, we shall, in this analysis, refer to the quantity as: $c_0^j = c_0^j(x^*)$. Thus,

$$c_0^j(x^*) = \int_{x^*}^{mid(\Delta^j)} \frac{1 - p(x)}{mid(\Delta^j) - \sigma^j} \mathrm{d}x + \int_{\sigma^j}^{x^*} \frac{p(x)}{mid(\Delta^j) - \sigma^j}\, \mathrm{d}x.$$

Let us now consider the function h defined by:

$h(x^*) = c_0^j(x^*) - c_1^j$,

where the reader should note that c_1^j does not depend on x^*. We will now examine the dynamics of $h(.)$ as x^* varies in the interval L^j (i.e., $\sigma^j \leq x^* < mid(\Delta^j)$).

To do this, we first consider the derivative of $h(\cdot)$ with regard to x^*. By evaluating this we can observe that:
$\frac{dh(x^*)}{dx^*} = \frac{dc_0^j(x^*)}{dx^*} = \frac{p(x^*)-1}{mid(\Delta^j)-\sigma^j} + \frac{p(x^*)}{mid(\Delta^j)-\sigma^j} = \frac{2p(x^*)-1}{mid(\Delta^j)-\sigma^j}$.
By utilizing the fact that $p(x) \geq 1/2$, we can conclude that $\frac{dh(x^*)}{dx^*} \geq 0$ for all x^*.

Since the function $h()$ is an increasing function, we shall proof that h is negative when x^* approaches the end of the interval ($h(\sigma^j) < 0$) and is positive when x^* approaches the other opposite end point ($h(mid(\Delta^j)) > 0$), which clearly demonstrates that h admits also a $zero$. This confirms that: $\exists\, x_0 \in L^j$ such that $h(x_0) = 0$, proving the result sought for.

Consider now:
$c_0^j(mid(\Delta^j)) = \int_{mid(\Delta^j)}^{\sigma^j} \frac{1-p(x)}{mid(\Delta^j)-\sigma^j}\,\mathrm{d}x.$
Similar to the proof of case analyzed earlier when $x^* \olessthan \Delta^j$, i.e, $x^* > \gamma^j$, we can follow analogous arguments to show that for $x^* = mid(\Delta^j)$, $c_1^j < c_0^j$ as in Case 2, implying that $h(mid(\Delta^j)) > 0$.

If $x^* = \sigma^j$, similar to the proof for $x^* \ogreaterthan \Delta^j$, we obtain that for $x^* = \sigma^j$, $c_0^j(\sigma^j) = \int_{\sigma^j}^{mid(\Delta^j)} \frac{p(x)}{mid(\Delta^j)-\sigma^j}\mathrm{d}x$, which reduces to Case 1 earlier studied, and we thus obtain that $c_0^j < c_1^j$. Thus, $h(\sigma^j) < 0$, and so $\exists x_0 \in L^j$ such that $h(x_0) = 0$.

The conclusion of this whole exercise is that the sign will change depending on wether x^* is closer to the left endpoint of the interval or to the right endpoint. Consequently, $Pr(\Omega^j = \{Left,\ Inside\ or\ Right\}) \to 1$, and the result is proven. $\square$

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

Proof:

First of all, we observe that:

$$\begin{aligned} x^\star \text{ in } \Delta \quad &\Rightarrow \quad \exists i \text{ such that } x^\star \ \in \Delta^i \\ &\Rightarrow \quad \exists i \text{ such that } x^\star \ominus LeftHalf(\Delta^i) or x^\star \ominus RightHalf(\Delta^i). \end{aligned}$$

We shall consider each of these cases separately.

Consider the case when $x^\star \ominus$ LeftHalf(Δ^i). Then,

$$\begin{aligned} &Pr(\Omega^j = Right) \to 1 \text{ for } j < i \\ &Pr(\Omega^i = \{Left,\ Inside\ or\ Right\}) \to 1 \\ &Pr(\Omega^j = Left) \to 1 \text{ for } j > i. \end{aligned}$$

We observe the following:

- If Ω^i= *Left*, the corresponding code is: $\vec{\Omega}_i$ for $1 \le i \le d$;
- If Ω^i = *Right*, the corresponding code is: $\vec{\Omega}_{i+1}$ for $1 \le i \le d$;
- If Ω^i = *Inside*, the corresponding code is: $\vec{\Omega}'_i$ for $1 \le i \le d$.

Let us now consider the case when $x^\star \ominus$ RightHalf(Δ^i). In this case, we have:

$$\begin{aligned} &Pr(\Omega^j = Right) \to 1 \text{ for } j < i \\ &Pr(\Omega^i = Right) \to 1 \\ &Pr(\Omega^j = Left) \to 1 \text{ for } j > i. \end{aligned}$$

It follows then that the corresponding code is $\vec{\Omega}_{i+1}$ for $1 \le i \le d$.

By considering the union of all symbols generated in both these cases, we observe that it reduces to the $2d+1$ entries stated in the theorem. □

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 3.2.*

Proof:

According to Theorem 2, the possible output vectors are defined by:

$$\{\vec{\Omega}_i, \text{such that}, 1 \le i \le d+1\} \bigcup \{\vec{\Omega}'_i, \text{such that}, 1 \le i \le d\}.$$

To achieve this proof, we will consider each of the $2d+1$ possible outputs, and thereafter conclusively infer that the new search-interval, Δ^{new} is guaranteed to always contain x^* (w. p. 1).

Case when the output vector is $\vec{\Omega}'_i$, $1 \le i \le d$: In this case, when the team of LA has converged to the output vector $\vec{\Omega}'_i$, $1 \le i \le d$, by considering Theorem 1, we can see that the only case where we observe the output vector $\vec{\Omega}'_i$, is whenever $x^\star \circledequal LeftHalf(\Delta^i)$. Thus, the pruning interval Δ^{new} corresponding to $\vec{\Omega}'_i$ is $LeftHalf(\Delta^i)$.

Case when the output vector is $\vec{\Omega}_i$, $1 \le i \le d+1$: To examine this case, let us first consider the scenario when $1 < i < d+1$ and consider the special cases where $i = 1$ and $i = d+1$ at the end of the proof.

When $1 < i < d+1$, the proof is rather complicated and is done in two steps. In the first step, we will prove, by contradiction, that if the output vector is $\vec{\Omega}_i$ then:

- It is impossible that x^* belongs to any of the intervals $\{\Delta^{i-2}$,Δ^{i-3},...$\Delta^1\}$. In fact, let us suppose that the team of LA has converged to $\vec{\Omega}_i$, and let suppose that $x^* \in \Delta^{i-2}$. The monotonicity of the intervals implies that $\Delta^{i-1} \circledless x^\star$. Thus, the $i-1^{th}$ component of $\vec{\Omega}_i$ is *Right*, leading to a contradiction. In a similar manner, we can prove, by contradiction, that it impossible that x^* belongs to any of the intervals $\{\Delta^{i-3}$,...., Δ^2,$\Delta^1\}$.

- It is impossible that x^* belongs to any of the intervals $\{\Delta^{i+1}$, Δ^{i+2},..., $\Delta^d\}$. In fact, let us suppose that $x^* \in \Delta^{i+1}$. As a consequence of the monotonicity of the search interval, $x^\star \circledless \Delta^i$. Thus the i^{th} component of $\vec{\Omega}_i$ is *Left*, leading again to a contradiction. A similar reasoning can be used to prove that it impossible that x^* belongs to any of the intervals $\{\Delta^{i+2}$, Δ^{i+2},..., $\Delta^d\}$.

We now approach the second step that follows after eliminating the intervals, $\{\Delta^{i-2}$,Δ^{i-3},...$\Delta^1\}$ and the intervals $\{\Delta^{i+2}$, Δ^{i+2}, ..., $\Delta^d\}$. We are now left with only two intervals, namely: Δ^{i-1} and Δ^i. To initiate this, we observe that for x^* belonging to either Δ^{i-1} or Δ^i, we have:

$$Pr(\Omega^j = Right) \rightarrow 1 \text{ for } 1 \ge j \le i-1,$$
$$Pr(\Omega^j = Left) \rightarrow 1 \text{ for } i+1 \ge j \le d.$$

We now consider all the possible positions of x^* relative to the intervals Δ^{i-1} and Δ^i. This leads us to the following four cases, which we will handle individually:

$x^* \in LeftHalf(\Delta^{i-1})$, $x^* \in RightHalf(\Delta^{i-1})$, $x^* \in LeftHalf(\Delta^i)$ and $x^* \in RightHalf(\Delta^i)$.

1. When $x^* \in LeftHalf(\Delta^{i-1})$, according to Theorem 1, $\Omega_{i-1} = Left$ and $\Omega_i = Left$, where the result that $\Omega_i = Left$ is a consequence of the monotonicity of the intervals Δ^{i-1} and Δ^i. Since $\Omega_{i-1} = Left \neq Right$, $x^* \in LeftHalf(\Delta^{i-1})$ does not lead to the output vector $\vec{\Omega}_i$.

2. $x^* \in RightHalf(\Delta^{i-1})$: Three possible output vectors could emerge according to the three possible convergence possibilities of Ω_{i-1}.

   - $\Omega_{i-1} = Left$ and $\Omega_i = Left$. Since $\Omega_{i-1} = Left \neq Right$, the current subcase does not lead to the output vector $\vec{\Omega}_i$.
   - $\Omega_{i-1} = Inside$ and $\Omega_i = Left$. Since $\Omega_{i-1} = Inside \neq Right$, the current subcase again does not lead to the output vector $\vec{\Omega}_i$.
   - $\Omega_{i-1} = Right$ and $\Omega_i = Left \Rightarrow$. This subcase leads to the corresponding output vector being $\vec{\Omega}_i$.

3. $x^* \in LeftHalf(\Delta^i)$. This means that $\Omega_{i-1} = Right$ and $\Omega_i = Left$, whence the corresponding output vector is $\vec{\Omega}_i$.

4. $x^* \in RightHalf(\Delta^i)$: As in the second case, three possible output vectors could emerge according to the three possible convergence possibilities for Ω_i.

   - $\Omega_{i-1} = Right$ and $\Omega_i = Left$. In this case, the corresponding output vector is $\vec{\Omega}_i$.
   - $\Omega_{i-1} = Right$ and $\Omega_i = Inside$. Since $\Omega_i = Inside \neq Right$, this subcase does not lead to the output vector $\vec{\Omega}_i$.
   - $\Omega_{i-1} = Right$ and $\Omega_i = Right$. In this case too, the corresponding output vector is $\vec{\Omega}_i$.

To summarize the above 4 cases with their corresponding subcases, for an output vector $\vec{\Omega}_i$, where $2 \leq i \leq d$, the only combinations that yield $\vec{\Omega}_i$ are:

- $x^{\star}$⊜RightHalf(Δ^{i-1}) and $Pr(\Omega^{i-1} = \{Right\}) \to 1$;
- $x^{\star}$⊜LeftHalf(Δ^{i-1}) and $Pr(\Omega^{i-1} = \{Right\}) \to 1$;
- $x^{\star}$⊜LeftHalf(Δ^{i}) and $Pr(\Omega^{i} = \{Left\}) \to 1$.

Thus, for an output vector $\vec{\Omega}_i$, where $2 \leq i \leq d$, the only possible locations of $x^{\star}$that would generate this output vector are $LeftHalf(\Delta^{i-1})$, $RightHalf(\Delta^{i-1})$ and $LeftHalf(\Delta^i)$. Thus:

$x^{\star} \in LeftHalf(\Delta^{i-1}) \cup RightHalf(\Delta^{i-1}) \cup LeftHalf(\Delta^i) \implies \Delta^{i-1} \cup LeftHalf(\Delta^i)$

By defining $\Delta^0 = \emptyset$ and $\Delta^{d+1} = \emptyset$, it is easy to extend the latter conclusion for the case where $i = 1$ and for the case where $i = d+1$. Thus, trivially, for $\vec{\Omega}_1$, the pruned entry is $\Delta^0 \cup LeftHalf(\Delta^1) = LeftHalf(\Delta^1)$. Similarly, for $\vec{\Omega}_{d+1}$, the pruned entry is $\Delta^d \cup LeftHalf(\Delta^{d+1}) = \Delta^d$.

Thus, we can conclude that, in general, for an output vector $\vec{\Omega}_i$, where $1 \leq i \leq d+1$:

$x^\star \in LeftHalf(\Delta^{i-1}) \cup RightHalf(\Delta^{i-1}) \cup LeftHalf(\Delta^i) \implies \Delta^{i-1} \cup LeftHalf(\Delta^i)$. □

Remark: Based on the above, it is easy to see that the decision tables given by Tables 1, 2 and 3, used to prune the search space of SRF–AdS for the cases when $d = 2$, $d = 3$ and $d = 4$, are complete.

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.*

Proof: The proof is straightforward. From the pruning table, we divide the initial interval into d sub-intervals. We continue to search in, at most, $\Delta^{i-1} \cup LeftHalf(\Delta^i)$ whenever the team of LA converges to the output vector $\vec{\Omega}_i$, which informally speaking represents "one-and-a-half" sub-intervals. Thus the SRF–AdS shrinks the search space by, at least, a factor of $\frac{d}{3/2} = \frac{2d}{3}$. □

We conclude this section by observing that althugh the rows of the decision table are the same as in the original approach, the pruning and decision rules for consistent responses are different.

5 Implementation and Evaluation of SRF–AdS Scheme

The SRF–AdS strategy is fairly simple to implement. This is because it uses a straightforward partitioning of the search interval, a simple decision table when it concerns the elimination, and the well known L_{RI} learning algorithm. In this section we present the pseudo-code for the overall learning strategy as well as that of the L_{RI} learning algorithm. We also present a sample trace (for $d = 3$) of how the algorithm runs so as to demonstrate the scheme's correct convergence.

5.1 Implementation of the SRF–AdS Strategy

The SRF–AdS strategy has been implemented and tested with a wide range of inputs. The pseudo-code for the algorithms are presented in presented in Figure 2.

A sample trace of how the algorithm operates is given in Figure 3, which also illustrates the workings of the SRF–AdS strategy when $d = 3$. In this example run, the initial search interval was $[-5, 5]$ and $x^\star$ was 0.9123. The search terminated when the width (i.e., the resolution) of the interval was less than a user-specified value. The reward factor θ of the automata was 0.8 and $\epsilon = 0.005$. In every invocation of SRF–AdS (for this value of d), the results of the automata are given as a set of decision outputs Ω^1, Ω^2 and Ω^3 and are determined after the L_{RI} scheme had run for $N_\infty = 250$ iterations. Note that at Step 10 in Figure 3, the algorithm terminated when the width of the interval [-0.2806, -0.2763] was less than the specified resolution (0.005). The estimated value for $x^\star$ was the mid-point of the interval $[-0.2792, -0.2770]$, which is -0.2781.

6 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

```
Algorithm SRF–AdS($\Delta^{Original}$)
Input : $p$, $\theta$, $\epsilon$, Resolution and $N_\infty$, all of which are assumed global.
Output : The final estimate $\hat{E}(x(N_\infty))$.

Method :
Begin
    $\Delta := [\sigma, \gamma)$; $\Delta^{Original} := [0, 1)$
    For $i := 1$ To $d$ Do
        $\Delta^i := [\frac{(i-1)(\gamma-\sigma)}{d}, \frac{i(\gamma-\sigma)}{d})$
    EndFor
    For $j := 1$ To $d$ Do
        $\Omega^j$ := Execute_$L_{RI}$($j$)
    EndFor
    $\vec{\Omega} := [\Omega^j, j = 1, 2, \ldots d]$
    $\Delta^{new}$ := ChooseNewSearchInterval($\vec{\Omega}$, Decision-Table)
END Algorithm SRF–AdS


Procedure Search($\Delta$)
Input : Resolution: the size of the smallest significant interval containing $x^\star$. Its magnitude determines the accuracy of the
        final estimate and is used to terminate the recursion. The function MidPointOfInterval returns the mid-point of the
        specified interval. Also, the function PartitionInterval partitions the given interval into $d$ sub-intervals. These are trivial,
        and are thus not described here.
Output : The estimate of $x^\star$. It is derived as the mid-point of the final search interval.

Method :
Begin
    If (WidthOfInterval($\Delta$) $\leq$ Resolution) Then
      Return (MidPointOfInterval($\Delta$))                                  /* Terminate Recursion */
    Else
      [$\Delta^1$, $\Delta^2$, $\ldots \Delta^d$] := PartitionInterval ($\Delta$)
      For $j := 1$ To $d$ Do
          $\Omega^j$ := Execute_$L_{RI}$($j$, EnvironType)
      EndFor
      $\vec{\Omega} := [\Omega^j, j = 1, 2, \ldots d]$
      $\Delta^{new}$ := ChooseNewSearchInterval($\vec{\Omega}$, Decision-Table)
      Search($\Delta^{new}$, EnvironType)                                    /* Tail Recursion */
    EndIf
END Procedure Search
```

Figure 2: Algorithm SRF–AdS

Procedure Execute_L_{RI}(j)

Input: The sub-interval, Δ^j; the parameters θ and ϵ of the L_{RI} scheme. The functions *ChooseAction*, *PickARandomPointIn* and *GetFeedBack* are trivial from a LA perspective, and are hence not explained in detail. If the user opts to use any other ϵ-optimal scheme, for example, from the family of estimator algorithms, he should replace the updating equations in this module. Also, the reader will observe that there is some duplication of statements in the "If-Then-Else" blocks. This is done just to improve the readability.

Output: A decision from the set {Left, Right, Inside} representing whether the automaton has determined $x^\star$ to be to the $Left$, $Right$ or $Inside$ the current partition.

```
Method :
Begin
    $P_0^j := P_1^j := 0.5$
    For $i$ := 1 To $N_\infty$ Do
        $k$ := ChooseAction($\Delta^j$)
        If ($k$ = 0) Then                                   /* $\alpha_0^j$ is the chosen action */
            $x_L^j$:= PickARandomPointIn($L^j$)             /* $L^j$ is the left half of $\Delta^j$ */
            $\beta$ := GetFeedBack(Y($x_L^j$))              /* Compare Y($x_L^j$) with 0
            If ($\beta$ = 0) Then                           /* Oracle has rewarded the choice */
                $P_1^j := \theta.P_1^j; P_0^j := 1 - P_1^j$
            EndIf
        Else                                                /* $\alpha_1^j$ is the chosen action */
            $x_R^j$:= PickARandomPointIn($R^j$)             /* $R^j$ is the right half of $\Delta^j$ */
            $\beta$ := GetFeedBack(Y($x_R^j$)               /* Compare Y($x_R^j$) with 0
            If ($\beta$ = 0) Then                           /* Oracle has rewarded the choice */
                $P_0^j := \theta.P_0^j; P_1^j := 1 - P_0^j$
            EndIf
        EndIf
    EndFor
    If ($P_0^j \geq 1 - \epsilon$) Then Return (Left) EndIf
    If ($P_1^j \geq 1 - \epsilon$) Then Return (Right) EndIf
    Return (Inside)
End Procedure Execute_$L_{RI}$
```

Procedure ChooseNewSearchInterval ($\vec{\Omega}$, Decision-Table)

Input: The Results of the L_{RI} Automata $\{\mathcal{A}^j\}$; Λ : The Decision-Table associated with the number of sub-intervals, d.

Output: The Δ^{new}, the new sub-interval to be processed.

```
Method
Begin
    If ($\vec{\Omega} \in$ Table $\Lambda$) Then
        Return (NewSubInterval(Table $\Lambda$, $\vec{\Omega}$))
    Else
        Return ($\Delta^{Original}$)
    EndIf
End Procedure ChooseNewSearchInterval
```

Step 1:$\Delta = [-5.0, 5]$
Partitions: $\Delta^1 = [-5.0, -1.6666]$ $\Delta^2 = [-1.6666, 1.6666]$ $\Delta^3 = [1.6666, 5.0]$
Results: $\Omega^1 = Right$ $\Omega^2 = Right$ $\Omega^3 = Left$
Conclusion: New Search Interval is $\Delta^2 \cup LeftHalf(\Delta^3) = [-5.0, 0]$

Step 2:$\Delta = [-5.0, 0]$
Partitions: $\Delta^1 = [-5.0, -3.3333]$ $\Delta^2 = [-3.333, -1.6666]$ $\Delta^3 = [-1.6666, 0]$
Results: $\Omega^1 = Right$ $\Omega^2 = Right$ $\Omega^3 = Right$
Conclusion: New Search Interval is $\Delta^3 = [-1.6666, 0]$

Step 3:$\Delta = [-1.6666, 0]$
Partitions:$\Delta^1 = [-1.6666, -1.1111]$ $\Delta^2 = [-1.1111, -0.5555]$ $\Delta^3 = [-0.5555, 0]$
Results: $\Omega^1 = Right$ $\Omega^2 = Right$ $\Omega^3 = Right$
Conclusion: New Search Interval is $\Delta^3 = [-0.5555, 0]$

Step 4:$\Delta = [-0.5555, 0]$
Partitions:$\Delta^1 = [-0.5555, -0.3703]$ $\Delta^2 = [-0.3703, -0.1851]$ $\Delta^3 = [-0.1851, 0]$
Results: $\Omega^1 = Right$ $\Omega^2 = Right$ $\Omega^3 = Left$
Conclusion: New Search Interval is $\Delta^2 \cup LeftHalf(\Delta^3) = [-0.3703, -0.0925]$

Step 5:$\Delta = [-0.3703, -0.0925]$
Partitions:$\Delta^1 = [-0.3703, -0.2777]$ $\Delta^2 = [-0.2777, -0.1851]$ $\Delta^3 = [-0.1851, -0.0925]$
Results: $\Omega^1 = Right$ $\Omega^2 = Left$ $\Omega^3 = Left$
Conclusion: New Search Interval is $\Delta^1 \cup LeftHalf(\Delta^2) = [-0.3703, -0.2314]$

Step 6:$\Delta = [-0.3703, -0.2314]$
Partitions:$\Delta^1 = [-0.3703, -0.3240]$ $\Delta^2 = [-0.3240, -0.2777]$ $\Delta^3 = [-0.2777, -0.2314]$
Results: $\Omega^1 = Right$ $\Omega^2 = Right$ $\Omega^3 = Left$
Conclusion: New Search Interval is $\Delta^2 \cup LeftHalf(\Delta^3) = [-0.3240, -0.2546]$

Step 7:$\Delta = [-0.3240, -0.2546$
Partitions:$\Delta^1 = [-0.3240, -0.3009]$ $\Delta^2 = [-0.3009, -0.2777]$ $\Delta^3 = [-0.2777, -0.2546]$
Results: $\Omega^1 = Right$ $\Omega^2 = Right$ $\Omega^3 = Left$
Conclusion: New Search Interval is $\Delta^2 \cup LeftHalf(\Delta^3) = [-0.3009, -0.2662]$

Step 8:$\Delta = [0.899, 0.929]$
Partitions:$\Delta^1 = [-0.3009, -0.2893]$ $\Delta^2 = [-0.2893, -0.2777]$ $\Delta^3 = [-0.2777, -0.2662]$
Results: $\Omega^1 = Right$ $\Omega^2 = Right$ $\Omega^3 = Left$
Conclusion: New Search Interval is $\Delta^2 \cup LeftHalf(\Delta^3) = [-0.2893, -0.2719]$

Step 9:$\Delta = [-0.2893, -0.2719]$
Partitions:$\Delta^1 = [-0.2893, -0.2835]$ $\Delta^2 = [-0.2835, -0.2777]$ $\Delta^3 = [-0.2777, -0.2719]$
Results: $\Omega^1 = Right$ $\Omega^2 = Right$ $\Omega^3 = Left$
Conclusion: New Search Interval is $\Delta^2 \cup LeftHalf(\Delta^3) = [-0.2835, -0.2748]$

Step 10:$\Delta = [-0.2835, -0.2748]$
Partitions:$\Delta^1 = [-0.2835, -0.2806]$ $\Delta^2 = [-0.2806, -0.2777]$ $\Delta^3 = [-0.2777, -0.2748]$
Results: $\Omega^1 = Right$ $\Omega^2 = Left$ $\Omega^3 = Left$
Conclusion: New Search Interval is $\Delta^2 \cup LeftHalf(\Delta^3) = [-0.2806, -0.2763]$

Step 11:$\Delta = [-0.2806, -0.2763]$
Partitions:$\Delta^1 = [-0.2806, -0.2792]$ $\Delta^2 = [-0.2792, -0.2777]$ $\Delta^3 = [-0.2777, -0.2763]$
Results: $\Omega^1 = Right$ $\Omega^2 = Left$ $\Omega^3 = Left$
Conclusion: New Search Interval is $\Delta^2 \cup LeftHalf(\Delta^3) = [-0.2792, -0.2770]$

Figure 3: Trace of the execution of an example run of SRF–AdS algorithm for the case when $d = 3$.

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:

$$g_1(x) = -9x + 3,$$

and

$$g_2(x) = exp(-5x) - 4.$$

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.

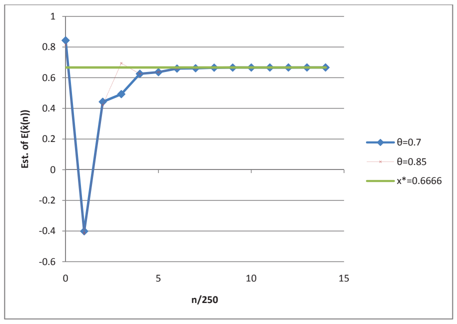
6.1 Experiments for Tertiary Search

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

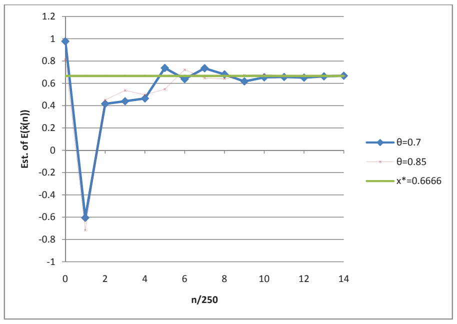
6.1.1 Linear Function

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 4(a), 4(b) and 4(c), where the standard deviations of the noise are $\sigma = 0.2$, $\sigma = 0.7$ and $\sigma = 1.0$ respectively.

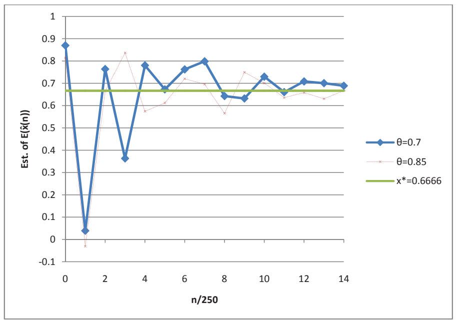
The reader must observe, first of all, that the algorithms converged to the true root *in every single case,* and that in every epoch, the *search space was decreased significantly*. One must also observe that as we increased the noise steadily from 0.2 to 1.0, the convergence speed decreased – which 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 4(a), Figure 4(b) and Figure 4(c).



(a)



(b)



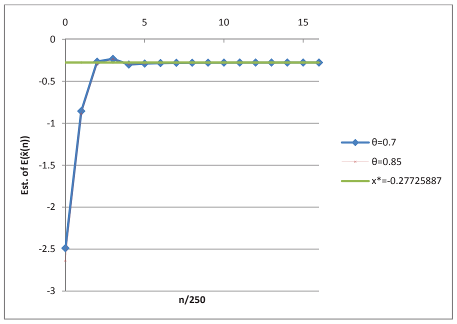
(c)

Figure 4: 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) $\sigma = 1.0$.

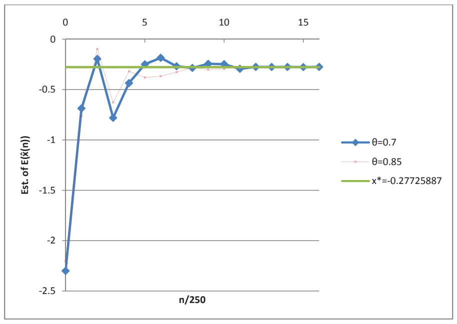
6.1.2 Non-Linear Function

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.

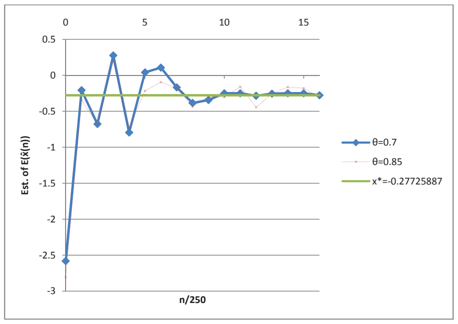
The variations are plotted n Figures 5(a), 5(b) and 5(c) for the scenarios when the standard deviations of the noise were respectively $\sigma = 0.2$, $\sigma = 0.7$ and $\sigma = 1.0$. Again, we observe that as expected, as we increased the noise steadily from 0.2, to 1.0, the convergence speed decreased. 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 5(a), 5(b) and 5(c). But in every case, one should note that SRF–AdS converged to the true but unknown root.



(a)



(b)



(c)

Figure 5: 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) $\sigma = 1.0$.

6.2 Effect of Increasing the Number of partitions d

We also did experiments to investigate the effect of increasing the number of partitions, d.

To study this, we varied d using the value which was earlier 3, from 5 to 8. We used the exponential function g_2, and the L_{RI} reward factor was fixed to $\theta = 0.85$. In Figure 6, we plot the variation of $\hat{E}[\hat{x}(n)]$ with time n when $\sigma = 0.2$ and with d varying as described above. The results shown in Figure 7 are for the case when $\sigma = 0.8$.

Both Figures 6 and 7 demonstrate a significant increase in the convergence speed as we increased the number of partitions. Further, our experiments show that our algorithm possesses the potential of being parallelized – which comes at the cost of increasing the computational cost.

6.3 Asymptotic Value

In the interest of completeness, we have also recorded the mean asymptotic values obtained in the various experiments. The mean asymptotic values of the estimate for $x^\star$ are shown in Table 5 for various values of σ and θ. As seen in the table, the final estimates agree with the true value $x_2^* = -Ln(4)/5 \approx -0.27725887$ of $x^\star$to the first two decimal places for *all* values of σ.

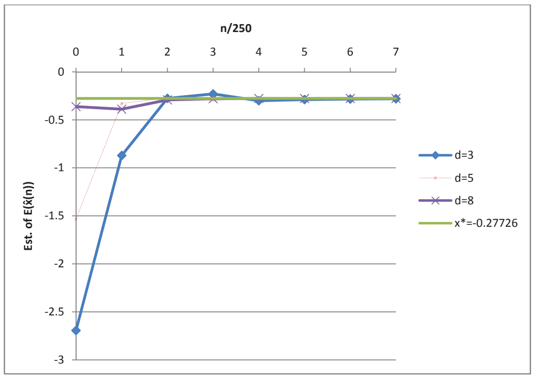


Figure 6: Variation of $\hat{E}[\hat{x}(n)]$ with time n, when the noise standard deviation $\sigma = 0.2$ and the L_{RI} reward factor $\theta = 0.85$, and the number of partitions d varies from 3 to 8. Here $x_2^* = -Ln(4)/5 \approx -0.27725887$. The time n is shown in terms of epochs of size 250 units.

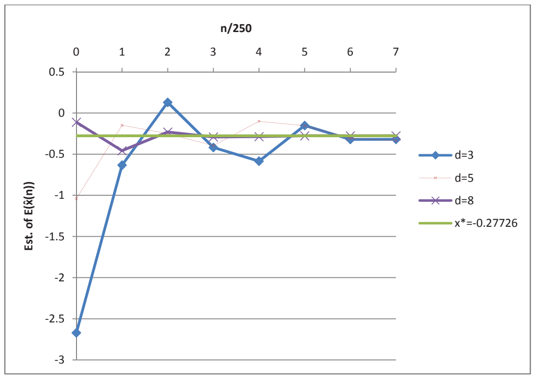


Figure 7: Variation of $\hat{E}[\hat{x}(n)]$ with time n, when the noise standard deviation $\sigma = 0.8$ and the L_{RI} reward factor $\theta = 0.85$, and the number of partitions d varies from 3 to 8. Here $x_2^* = -Ln(4)/5 \approx -0.27725887$. The time n is shown in terms of epochs of size 250 units.

σ	$\theta = 0.8$	$\theta = 0.85$	$\theta = 0.9$
0.10	-0.277269	-0.277262	-0.277258
0.30	-0.277241	-0.277249	-0.277251
0.50	-0.277212	-0.277220	-0.277231
0.70	-0.276129	-0.276145	-0.276212
0.80	-0.275437	-0.275743	-0.275998
1.0	-0.275126	-0.275145	-0.275638

Table 5: The asymptotic values of $\hat{E}[x(\infty)]$ for various values of σ and θ when SRF-AdS was invoked with $d = 3$. In all the cases, $x^* = -0.27725887$, $N_\infty = 250$ and $\epsilon = 0.005$. The values shown are averaged over an ensemble of 50 independent experiments.

7 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 [15]. 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 [9], and in particular to the Continuous Point Location with Adaptive d-ary Search (CPL–AdS), originally presented in [13]. 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.* [21] 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 [21] is an order of magnitude faster than classical SPL solution [9]; 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] S. Lakshmivarahan. *Learning Algorithms Theory and Applications*. Springer-Verlag, New York, 1981.

[6] 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.

[7] K. S. Narendra and A. M. Annaswamy. *Stable Adaptive Systems*. Prentice-Hall, New Jersey, 1989.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

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

[16] Y. Sawaragi and N. Baba. A note on the learning behavior of variable-structure stochastic automata. *IEEE Transactions on Systems, Man and Cybernetics*, 3:644–647, 1973.

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

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

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

[20] 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.

[21] 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.