Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games (CIG 2007)

Using Stochastic AI Techniques to Achieve Unbounded Resolution in Finite Player Goore Games and its Applications

B. John Oommen[1]
School of Computer Science
Carleton University
Ottawa, Ontario, Canada
oommen@scs.carleton.ca

Ole-Christoffer Granmo
Department of ICT
Agder University College
Grimstad, Norway
ole.granmo@hia.no

Asle Pedersen
Agder ICT Center
Grimstad, Norway
asle.pedersen@sts.no

***Abstract*— The Goore Game (GG) introduced by M. L. Tsetlin in 1973 has the fascinating property that it can be resolved in a completely distributed manner with no inter-communication between the players. The game has recently found applications in many domains, including the field of sensor networks and Quality-of-Service (QoS) routing. In actual implementations of the solution, the players are typically replaced by Learning Automata (LA). The problem with the existing reported approaches is that the accuracy of the solution achieved is intricately related to the *number* of players participating in the game – which, in turn, determines the resolution. In other words, an arbitrary accuracy can be obtained only if the game has an *infinite* number of players. In this paper, we show how we can attain an unbounded accuracy for the GG by utilizing no more than *three* stochastic learning machines, and by recursively pruning the solution space to guarantee that the retained domain contains the solution to the game with a probability as close to unity as desired. The paper also conjectures on how the solution can be applied to some of the application domains.**

Keywords: Learning Automata, Intelligent Game Playing, Goore Games, Sensor Networks and Quality-of-Service Routing

I. Introduction

One of the most fascinating games studied in the field of artificial games is the Goore Game[2] (GG) described below using the informal formulation of [1].

"*Imagine a large room containing N cubicles and a raised platform. One person (voter) sits in each cubicle and a Referee stands on the platform. The Referee conducts a series of voting rounds as follows. On each round the voters vote "Yes" or "No" (the issue is unimportant) simultaneously and independently (they do not see each other) and the Referee counts the fraction, λ, of "Yes" votes. The Referee has a uni-modal performance criterion $G(\lambda)$, which is optimized when the fraction of "Yes" votes is exactly λ^*. The current voting round ends with the Referee awarding a dollar with probability $G(\lambda)$ and assessing a dollar with probability $1 - G(\lambda)$ to every voter independently. On the basis of their individual gains and losses, the voters then decide, again independently, how to cast their votes on the next round.*"

The game has many interesting and fascinating features which render it both non-trivial and "intriguing". These are listed below:

1) The game is a non-trivial *non-zero-sum* game.
2) Unlike the games traditionally studied in the AI literature (like Chess, Checkers, Lights-Out etc.) the game is essentially a *distributed* game.
3) The players of the game are ignorant of all of the game's "parameters". All they know is that they have to make a choice, for which they are either rewarded or penalized. They have no clue as to how many other players there are, how they are playing, or even of how/why they are rewarded/penalized.
4) The stochastic function used to reward or penalize the players can be completely arbitrary, as long as it is uni-modal.
5) The most "intriguing feature" of this game [1] is that if each voter updates its action based on either a Tsetlin automaton with large memory, or an absolutely expedient[3] algorithm, then the entire group will asymptotically optimize the Referee's performance criterion.

A. Applications to the Goore Game

The literature concerning the GG is scant. It was initially studied in the general learning domain, and, as far as we know, was for a long time merely considered as an interesting pathological game. Recently, however, the GG has found important applications within two main areas, namely, QoS (Quality of Service) support in wireless sensor networks [7] and within cooperative mobile robotics as summarized in [8]. A description of these two application areas follows.

In order to preserve energy in a battery driven network, Iyer *et al.* [9] proposed a scheme where a base station provided broadcasted QoS feedback to the sensors of the

[1] *Chancellor's Professor* ; *Fellow : IEEE* and *Fellow : IAPR*. The Author also holds an *Adjunct Professorship* with the Dept. of ICT, Agder University College, Norway.

[2] This game is also referred to as the *Gur Game* in the related literature.

[3] We assume that the reader has a fairly fundamental knowledge of the field of Learning Automata. Excellent reviews of this material can be found in [1]–[6].

network. Using the GG perspective, a sensor is seen as a voter that chooses between transmitting data or remaining idle in order to preserve energy. Correspondingly, the base station takes the role of the Referee and rewards/punishes the sensors using a unimodal QoS performance criterion function with the goal of attaining an optimal resolution/energy-usage trade-off.

Furthermore, Tung and Kleinrock [10] have demonstrated how the GG can be used for coordinating groups of mobile robots (also called "mobots") that have a restricted ability to communicate. The mobots [10] can either (1) collect pieces of ore from a landscape, or (2) sort already collected ore pieces. The mobots vary with respect to how fast they collect and sort these pieces of ore. The GG is used to make sure that the mobots optimally choose their actions.

Other possible cooperative robotics applications include controlling a moving platform and guarding a specified perimeter [8]. In all of these cases, the solution to the problem in question would essentially utilize the solution to the GG in a plug-and-play manner.

B. Known LA Solutions to the Goore Game

Learning Automata (LA) [1]–[6] have been used to model biological learning systems and to find the optimal action that is offered by a random Environment. Learning is accomplished by explicitly interacting with the Environment and processing its responses to the actions that are chosen, while gradually converging toward an ultimate goal. LA have found various applications in the past two decades. The learning loop involves two entities, the Random Environment and a Learning Automaton. A complete study of the theory and applications of LA can be found in excellent books by Lakshmivarahan [2], Narenda and Thathachar [1], Najim *et al.* [3] and Poznyak *et al.* [5]. Besides these, a recent issue of the *IEEE Transactions on Systems, Man and Cybernetics* [4] (also see [11]), has been dedicated entirely to the study of LA, and a more recent book [6] describes the state of the art when it concerns networks and games of LA. Some of the fastest reported LA belong to the the family of estimator algorithms whose study was initiated by Thathachar and Sastry, and followed by others [4], [12], [13].

We assume that we are dealing with a "team" of d LA, $\{A^1, A^2, \ldots A^d\}$. In terms of notation, we assume that the actions offered to each LA, A^j, from the Environment in question are $\{\alpha_0^j, \alpha_1^j\}$, and that the corresponding penalty probabilities are $\{c_0^j, c_1^j\}$ respectively. Similarly, we let $P_k^j(n)$ represent the component of the action probability vector of A^j for action α_k^j, where n represents the discretized time index.

In the interest of simplicity, throughout this paper, we shall assume that the individual LA used is the well-known L_{RI} scheme with parameter θ [1]–[6]. Any other absolutely expedient (or probably ϵ-optimal scheme – including those belonging to the estimator families cited above) can be used just as effectively. Thus, we first state a fundamental result for the L_{RI} learning scheme 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.* □

The above result is well known [1], [2]. Thus, 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 $P_k^j(n) \to 1$ as $n \to \infty$ and $\theta \to 1$.

II. Fundamentals of the Goore Game

Let $G(\cdot)$ be an arbitrary uni-modal function from $[0, 1] \to [0, 1]$ known to the Referee interacting with a team of d LA, $\{A^1, A^2, \ldots A^d\}$. Each LA, A^j, independently chooses an action $\alpha^j(n)$ which is either α_0^j or α_1^j, for which it receives, from the Referee, a response $\beta^j \in \{0, 1\}$, (with $\beta^j = 0$ being as a *Reward*) as per:

$$\beta^j(n) = 0 \qquad \textbf{w.p.} \qquad G\left(\frac{\sum_{j=1}^{d} \alpha^j(n)}{d}\right). \quad (1)$$

We now state the fundamental LA - Goore Game (LA-GG) Property.

Theorem 1. LA-GG Property

If each LA, A^i, receives its feedback signals from the Referee as per Equation (1), and uses the L_{RI} learning scheme with parameter $0 \ll \theta < 1$ to update its learning model, then each LA converges so that the collective behavior of the team optimizes the unknown function $G(\cdot)$. Thus, if $k^+ = \lim_{n\to\infty} \sum_{j=1}^{d} \alpha^j(n)$,

$$G\left(\frac{k^+}{d}\right) > G\left(\frac{k}{d}\right) \qquad \forall \quad k \neq k^+. \quad (2)$$

Proof: The is essentially a fundamental result [1], [14] whose proof is omitted.

Example 1.

Let us suppose that the function $G(\cdot)$ used by the Referee is $G(x) = 0.7 \times e^{-\frac{(0.9123-x)^2}{0.0625}}$ and that 5 LA are participating in the game. Then, the number of LA who vote "Yes" could be in the set $\{0, 1, 2, 3, 4, 5\}$, with their corresponding G values : $G(0) = 0.000015$, $G(0.2) = 0.000208$, $G(0.4) = 0.01050$, $G(0.6) = 0.14702$, $G(0.8) = 0.57209$, $G(1.0) = 0.61895$.

Observe that although $G(\cdot)$ has its maximum value at 1.0 within the *discretized* domain, the maximum of the function itself occurs at 0.9123. Theorem 1 claims that if each LA is an L_{RI} scheme with parameter θ being arbitrarily close to unity, *all* of the 5 LA will converge to a "Yes" vote (i.e., to a value of $\alpha = 1$) with a probability as close to unity as desired. □

A. Problems with reported LA solutions to the GG

The above solution to the GG is indeed both intriguing and actually, almost "mystical". Without knowledge of the function $G(\cdot)$, of how their partners decide, or even a perception of how the Referee "manufactured" their responses, the LA converge to the optimal solution *without* any communication. However, the main handicap associated with using it in

real-life applications concerns the *accuracy* of the solution obtained, which is intricately linked to the number of LA used. If the number of LA involved in the game is d, the precision of the solution is bounded by $\frac{1}{d}$, and thus the solution can be arbitrarily accurate only as d is increased indefinitely - leading to extremely slow convergence.

B. Salient Aspects of the Paper

The contributions of the paper are the following:

1) We report the first solution to the GG which needs only a finite number of LA. Indeed, the number of LA can be as small as 3.
2) We report the first GG solution which is arbitrarily accurate.
3) The solution we propose is recursive. To the best of our knowledge, there has been no other reported recursive solution.
4) The solution that we propose is "fast". Although this is a relative term, it turns out that, usually, each epoch of the recursion converges within a few hundred iterations, and the accuracy of the solution increases exponentially with the number of recursive calls. It is thus, arguably, the first reported realistic solution to this intriguing game.

The problem we study is akin to the ones studied in [15]–[19] for, the point location problem. The solution we propose is related, in principle, to the tertiary and d-ary recursive search mechanisms earlier proposed for the stochastic version of the latter [16], [17], [19]. But unlike the solutions reported in [16], [17], [19], the solution here is far more consequential because the system does not rely on a Teacher or "Oracle" instructing the LA which way it should move. Thus, our solution will have applications in all the areas mentioned earlier for which the GG has found *direct* applications [9], [10], and for the areas where the entire field of LA and stochastic learning, has found uni-modal optimization applications from a finite or infinite action set [1]–[3], [5], [6], [18], [20].

III. Continuous Goore Game with Adaptive d-ary Search

The solution presented in this paper is based on a strategy, the so-called Continuous Goore Game with Adaptive d-ary Search (CGG–AdS) strategy. The basic idea behind the CGG–AdS solution is to use d LA to play the GG, and then to use the results of *their* solution to systematically explore a *sub*-interval of the current interval for the solution. This exploration is a series of estimates, each one more accurate than the previous one.

In CGG–AdS, the given search interval is divided into d partitions representing d disjoint sub-intervals. In each interval, the LA utilize their distributed learning capabilities (with no mutual communication) to attain a consensus as to where the optimal point lies. Based on the collective response, the Referee then prunes the space *without informing the LA*, and eliminates at least one of the sub-intervals from being searched further. The search is then recursively invoked within the remaining pruned contiguous interval until the search interval is at least as small as the required resolution of estimation. This elimination process essentially utilizes the ϵ-optimality property of the underlying automata and the monotonicity of the intervals to guarantee the convergence, as stated in Theorem 1.

A. Notations and Definitions

Let $\Delta(t) = [\sigma, \gamma)$ s.t. $\sigma \leq \lambda^\star < \gamma$ be the current search interval at epoch t, containing $\lambda^\star$ whose left and right (smaller and greater) boundaries on the real line are σ and γ respectively. $\Delta(0)$ is initialized to be the unit interval. We partition $\Delta(t)$ into d equi-sized disjoint partitions[4] Δ^j, $j \in \{1, 2, \ldots d\}$, such that, $\Delta^j = [\sigma^j, \gamma^j)$. To formally describe the relative locations of 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 monotonic, $\Delta^1 \prec \Delta^2 \ldots \prec \Delta^d$. Also, we say that

$$\lambda^\star \circledcirc \Delta^j \quad \text{iff} \quad \sigma^j \leq \lambda^\star < \gamma^j.$$

B. Construction of the Learning Automata

We associate d 2-action L_{RI} automaton $\{\mathcal{A}^j = (\Sigma^j, \Pi^j, \beta^j, \Upsilon^j, \Omega^j)\}$ where, Σ^j is the set of actions - representing "Yes" or "No" decisions, Π^j is the set of action probabilities, β^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 Referee has, in its possession, a secret arbitrary uni-modal function $G(\cdot)$ from $[0, 1] \rightarrow [0, 1]$. The Environment, E , for each LA, is governed by the response of the Referee, who, unknown to the LA, rewards or penalizes them based on the values of the function $G(\cdot)$ within the current interval of interest. It, in a distributed manner, governs the overall search strategy by providing the responses to the LA, and additionally enhancing CGG–AdS, by using a Pruning Decision Rule (PDR)[5], Λ, to prune the search interval. It achieves this by utilizing the LA-GG property and the decisions, Ω^j, made in the previous epoch by the d LA. Thus $\mathcal{A}^j$, $j \in \{1, \ldots d\}$, together with E and Λ completely define the CGG–AdS strategy. These are formalized below.

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 the LA casting a "No" vote, and α_1^j corresponds to the LA casting a "Yes" vote.
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*: (β^j)

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

[5]This rule is also referred to as the Pruning Table.

Each LA receives, from the Referee, a response $\beta^j \in \{0,1\}$, (with $\beta^j = 0$ being regarded as a *Reward*) as per Equation (1) given below:

$$\beta^j(n) = 0 \qquad \textbf{w.p.} \qquad G\left(\frac{\sum_{j=1}^{d}\alpha^j(n)}{d}\right).$$

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. Typically, N_∞ is chosen so as to ensure (with a very high probability) that the team of automata will have converged[6]. Ω^j indicates that the particular LA, $\mathcal{A}^j$, has decided that it should vote either "Yes" or "No" with an arbitrary high accuracy. The set of values that Ω^j can take and the preconditions are given by:

$$\Omega^j = \begin{cases} No & \text{If} \quad P_0^j(N_\infty) \geq 1-\epsilon. \\ Yes & \text{If} \quad P_1^j(N_\infty) \geq 1-\epsilon. \end{cases}$$

6) *The Pruning Decision Rule (PDR) for pruning the search space*: (Λ)
Since the actions chosen by each LA can lead to one of the two possible decisions, namely Yes or No, the number of "Yes" votes can be any integer in the set $\{0,1,\ldots,d\}$. Once the team of automata have made a decision regarding where they reckon $\lambda^\star$ to be (by virtue of their votes), the CGG–AdS reduces the size of the search interval by a factor of at least $\frac{2}{d}$. If k^+ is the number of "Yes" votes received, the new pruned search interval, Δ^{new}, for the subsequent learning phase (epoch) is generated according to the PDR, Λ, for the specific value of d, defined as follows:

$$\Delta(t+1) = \begin{cases} \Delta^1 & \text{If} \quad k^+ = 0. \\ \Delta^m \cup \Delta^{m+1} & \text{If} \quad k^+ = m; m \neq 0, d. \\ \Delta^d & \text{If} \quad k^+ = d. \end{cases} \tag{3}$$

The PDR (generally, synonymously and equivalently given as a table), Λ, is shown in Table I and for the case when $d = 3$ and in [20] for $d = 4$. Clearly, the table "prunes" the size of the interval, because $\Delta(t+1)$ at the next epoch is, at most, of length $\frac{2}{d}$.

We shall now derive the fundamental properties of CGG–AdS.

[6]This is always guaranteed if we use an absolutely expedient scheme in which the unit vectors are absorbing barriers [1]–[6].

TABLE I

THE DECISION TABLE, (Λ), TO PRUNE THE SEARCH SPACE OF CGG–AdS FOR $d = 3$ BASED ON THE LA OUTPUTS Ω^j. k^+ IS THE NUMBER OF LA WHO VOTE "YES".

k^+	**New Sub-interval : Δ^{new}**
0	Δ^1
1	$\Delta^1 \cup \Delta^2$
2	$\Delta^2 \cup \Delta^3$
3	Δ^3

IV. CONVERGENCE PROPERTIES OF CGG–AdS

We consider here the convergence results concerning CGG–AdS for the general GG. Lemmas 2 and 3 essentially use the ϵ-optimality property of L_{RI} automata to prove that they produce the correct decision output for each partition. These are then used in Theorem 2 to prove that the formula used to create the PDR is correct. This, thus, establishes that after elimination of one or more partitions, the remaining interval still contains $\lambda^\star$w. p. 1., thereby assuring convergence.

Lemma 2. *Consider an arbitrary GG with a Referee providing responses as per Equation (1), and the LA working with an L_{RI} scheme with a parameter θ which is arbitrarily close to unity. Then, for $1 \leq m \leq d$, the following is true:*

$$\textit{If } (\lambda^* \in \Delta^m), \textit{Then } Pr\left[(k^+ = m-1) \textit{ or } (k^+ = m)\right] \to 1.$$

Proof: The proof is found in [20] and omitted here in the interest of brevity.

Lemma 3. *Consider an arbitrary GG with a Referee providing responses as per Equation (1), and the LA working with an L_{RI} scheme with a parameter θ which is arbitrarily close to unity. Then the following is true:*

$$\begin{array}{ll} \textit{If } (k^+ = 0), & \textit{Then } Pr\left[(\lambda^* \in \Delta^1)\right] \to 1. \\ \textit{If } (k^+ = m, m \neq 0, d), & \textit{Then } P[(\lambda^* \in \Delta^m) \\ & \quad \textit{or } (\lambda^* \in \Delta^{m+1})] \to 1. \\ \textit{If } (k^+ = d), & \textit{Then } Pr\left[(\lambda^* \in \Delta^d)\right] \to 1. \end{array}$$

Proof: The proof of this result too is found in [20].

Theorem 2. *Consider an arbitrary GG with a Referee providing responses as per Equation (1), and the LA working with an L_{RI} scheme with a parameter θ which is arbitrarily close to unity. Then:*

1) *The rules specified in Section 2.2 defining the construction of the PDR is valid.*
2) *The search domain for the solution of the GG reduces at each step of the recursion by a factor of at least $\frac{2}{d}$.*
3) *The unknown $\lambda^\star$ is always contained (w. p. 1) in the new search-interval Δ^{new} resulting from the application of the PDR specified in Section 2.2.*

Proof: The proof invokes the above Lemmas and further relies on the transitivity of the regions $\{\Delta^i\}$. It is detailed in [20].

With these results, we are ready to construct a mechanism that can learn the optimal solution $\lambda^\star$ for the GG.

V. Implementation and Evaluation of CGG–AdS Scheme

The CGG–AdS strategy is fairly simple to implement, because it uses a straightforward partitioning of the search interval, a simple decision table for elimination, and the well known L_{RI} learning algorithm for playing the Goore Game. In this Section we present the pseudo-code for the overall learning strategy as well as that of the L_{RI} automata playing it. We also present in [20] a sample trace (for $d = 3$) to demonstrate its correct convergence. Finally, we present numerical results to augment our analysis presented in the previous sections.

A. Implementation of the CGG–AdS Strategy

The CGG–AdS strategy has been implemented and tested with a wide range of inputs. The pseudo-code for the algorithms (in Figure VI) is included in the Appendix, and sample traces are presented in [20] (omitted here in the interest of brevity) to illustrate the workings of the CGG–AdS strategy, where the latter is given for the case when $d = 3$.

The pseudo-code in Figure VI (in the Appendix) shows the recursive organization of the search, including the systematic pruning of the search interval. Each pruning decision is obtained by consulting Table I, after observing the outcome of an L_{RI} GG that has been projected into the *current* search interval. The algorithm is then recursively invoked. The recursion is terminated when the width of the interval is less than twice the desired accuracy. Indeed, it is the projection of the L_{RI} solution to the GG into increasingly smaller search intervals that allows unbounded solution precision.

Although we have done numerous experiments, we present here two specific examples, to highlight two crucial issues. In the experiments which we report, we used a Gaussian performance criterion function $G(\lambda) = ae^{-(\frac{\lambda^*-\lambda}{b})^2}$, allowing the magnitude and peakedness of the criterion function to be controlled by the parameters a and b respectively. This permitted us to simulate a wide variety of environments.

In the first experiment which we report, we considered the case when $G(\lambda)$ attains it maximum at 0.9123 - which was exactly the solution for the example given in [19]. This was done to highlight the difference between our recursive GG solution, and the solution presented earlier for the stochastic point location problem. Although the solutions reported in [16], [17], [19] were novel (and in the case of [19], it still remains the only known solution) the LA solution to the GG presented here do not have the luxury of a Teacher/"Oracle" to assist them. Secondly, each LA in the case of the results of [17], [19] have 3 possible decisions, and thus the size of the possible set of decisions is 3^d (a lot of information, indeed!!) – which is significantly reduced by the pruning to $O(d)$. Here, the number of possible solutions is significantly less – merely $O(d)$, and the reduction that the pruning can achieve is even less significant. Finally, and most importantly, each LA in [17], [19] has the advantage of knowing that if the solution is likely to be to the "Left" of a certain region Δ^i, it is even more likely to be to the left of a region Δ^j, where $j > i$. Our current solution has to infer all this - and that in a distributed manner - without knowing how their partners performed or how and why they got a penalty/reward.

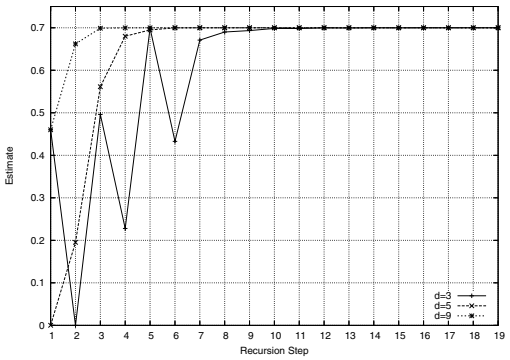

Fig. 1. Convergence of estimates for $d = 3, 5, 9$. The unknown parameter $\lambda^\star = 0.3139$, and the optimal value of the criterion function is 0.7.

The trace given in [20] shows the execution of the CGG–AdS algorithm for the case when $d = 3$. In this example run, the initial search interval is $[0, 1)$ and $\lambda^\star$ is 0.9123, and the parameters a and b were set to 0.7 and 0.035 respectively - which means that the optimal value of $G(\lambda)$ is 0.7. The search terminated when the width (i.e., the resolution) of the interval was ≤ 0.0002. The reward factor θ of the automata was 0.9999 and $\epsilon = 0.05$. In every invocation of CGG–AdS, the results of the automata are given as the optimal number of "Yes" votes, k^+. We remark that at Step 18 in the recursion, the width of the interval $[0.9121, 0.9123]$ is 0.0002, at which point the estimated value for $\lambda^\star$ is the mid-point of the interval $[0.9121, 0.9123]$, which is 0.9122. We note that at this resolution, our scheme is very close to optimizing the performance criterion function because $G(0.9122) \approx 0.69999$. The corresponding problem in the case of the solution in [19] converged after 10 recursive steps. It should be mentioned that the traditional LA solution to the GG would require $10,000$ LA to attain this level of precision. Hence, the power of our strategy !! Additional examples and traces for other executions of the solution are given in [20], and omitted here in the interest of space.

The analogous results for the second example are also given in [20] and illustrated in Figure 1.

We first note that as the solution resolution increases at each recursion step, the accuracy of the $\lambda^\star$estimates does not increase monotonically, as, perhaps, could have been expected. Instead, the estimates fluctuate noticeably, however, with decreasing variance. As argued in [20], this fluctuation is not a result of the random properties of our algorithm. However, for larger number of automata, the positioning of the sub-partitions seems to become less significant, as seen in Figure 2 for $d = 9$.

The reader should also note that at any given recursion step, the speed of convergence seems to be determined by

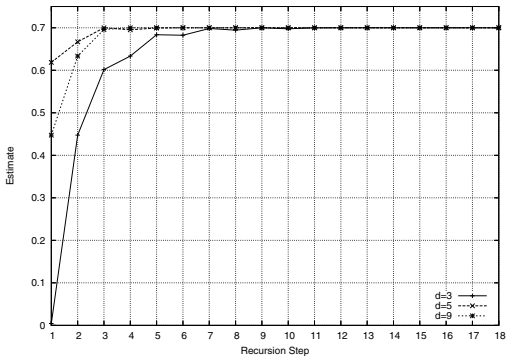

Fig. 2. Convergence of estimates for $d = 3, 5, 9$. The unknown parameter $\lambda^\star = 0.9123$, and the optimal value of the criterion function is 0.7.

the magnitude that the best available estimate λ^+ differs from the inferior estimates. Thus, for instance, a function G with a $G(\lambda)$ that is flat around the optimal value λ^* may slow down convergence when the search interval falls within the flat area. However, such a flat search interval means that all of the candidate estimates are close to the optimal value of the performance criterion function, and accordingly, the search could be terminated without a significant loss of accuracy.

VI. Conclusions

In this paper, we have considered a intriguing game, called the Goore Game (GG) introduced in [21], and which has recently found applications in many domains, including the field of sensor networks and Quality-of-Service (QoS) routing. The GG has the fascinating property that it can be resolved in a completely distributed manner with no inter-communication between the players. The existing reported approaches have a fundamental "ailment" : The accuracy of the solution achieved is intricately related to the *number* of players (typically, Learning Automata (LA)) participating in the game – which, in turn, determines the resolution. In other words, an arbitrary accuracy can be obtained only if the game has an *infinite* number of players, and thus a practical solution is infeasible. In this paper, we showed how we can attain an unbounded accuracy for the GG by utilizing at most d LA, and by recursively pruning the solution space to guarantee that the retained domain contains the solution to the game with a probability as close to unity as desired. Indeed, d can be made as small as *three*. The paper contains the formal algorithms, the claims of the respective convergence results, and it includes simulation results demonstrating its power. Indeed, we believe that we have presented here the first *practical* implementation of the GG.

We are currently investigating the application of these results to a variety of potential applications involving neural networks and optimization, and in the application domains related to sensor networks and QoS routing.

References

[1] K. Narendra and M. Thathachar, *Learning Automata*. Prentice-Hall, 1989.

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

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

[4] M. S. Obaidat, G. I. Papadimitriou, and A. S. Pomportsis, "Learning automata: Theory, paradigms and applications," *IEEE Transactions on Systems Man and Cybernetics*, vol. SMC-32, pp. 706–709, 2002.

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

[6] M. A. L. T. Thathachar and P. S. Sastry, *Networks of Learning Automata : Techniques for Online Stochastic Optimization*. Boston: Kluwer Academic, 2003.

[7] D. Chen and P. K. Varshney, "QoS Support in Wireless Sensor Networks: A Survey," in *The 2004 International Conference on Wireless Networks (ICWN 2004)*, 2004.

[8] Y. U. Cao, A. S. Fukunaga, and A. Kahng, "Cooperative Mobile Robotics: Antecedents and Directions," *Autonomous Robots*, vol. 4, no. 1, pp. 7–27, 1997.

[9] R. Iyer and L. Kleinrock, "Qos control for sensor networks," in *IEEE International Conference on Communications*, vol. 1, 2003, pp. 517–521.

[10] B. Tung and L. Kleinrock, "Using Finite State Automata to Produce Self-Optimization and Self-Control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, no. 4, pp. 47–61, 1996.

[11] M. S. Obaidat, G. I. Papadimitriou, and A. S. Pomportsis, "Effi cient fast learning automata," *Information Sciences*, vol. 157, pp. 121–133, 2003.

[12] 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*, vol. SMC-32(B), pp. 738– 749, 2002.

[13] J. Lancôt and B. Oommen, "Discretized estimator learning automata," *IEEE Transactions on Systems Man and Cybernetics*, vol. SMC-22, pp. 1473– 1483, 1992.

[14] M. A. L. Thathachar and M. T. Arvind, "Solution of Goore game using models of stochastic learning automata," *J. Indian Institute of Science*, no. 76, pp. 47–61, January-February 1997.

[15] R. A. Baeza-Yates, J. C. Culberson, and G. J. E. Rawlins, "Searching with uncertainty," in *Proceedings of the Scandinavian Workshop on Algorithms and Theory, SWAT 88*, 1988, pp. 176–189.

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

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

[18] G. Santharam, P. Sastry, and M. Thathachar, "Continuous action set learning automata for stochastic optimization," *Journal of the Franklin Institute*, vol. 331B5, pp. 607–628, 1994.

[19] B. J. Oommen, G. Raghunath, and B. Kuipers, "Parameter learning from stochastic teachers and stochastic compulsive liars," *IEEE Transactions on Systems Man and Cybernetics*, p. To Appear, 2006.

[20] B. J. Oommen, O. C. Granmo, and A. Pedersen, "Achieving unbounded resolution in *fi nite* player goore games using stochastic automata, and its applications," *Unabridged version of this paper. Submitted for publication*, 2006.

[21] M. Tsetlin, *Automaton Theory and the Modelling of Biological Systems*. New York and London: Academic Press, 1973.

Appendix

Program Search(Δ)
Input : Δ: Search interval $[\sigma, \gamma)$ containing $\lambda^\star$. *Resolution:* The size of the smallest significant interval containing λ^*. The function *MidPointOfInterval* returns the mid-point of the specified interval and the function *PartitionInterval* partitions the given interval into d sub-intervals.
Output The estimate of $\lambda^\star$.

Method :
Begin
  If (WidthOfInterval(Δ) $\leq$ Resolution) **Then**
    Return (MidPointOfInterval(Δ)) /* Terminate Recursion */
  Else
    $\{\Delta^0, \ldots, \Delta^d\}$:= PartitionInterval(Δ)
    k^+ := ExecuteGooreGame($\{\Delta^0, \ldots, \Delta^d\}$)
    Δ^{new} := ChooseNewSearchInterval($\{\Delta^0, \ldots, \Delta^d\}$, k^+, Decision-Table)
    Search(Δ^{new}) /* Tail Recursion */
  EndIf
END Program Search

Procedure ExecuteGooreGame($\{\Delta^0, \ldots, \Delta^d\}$)
Input : The partitioned search interval Δ= $[\sigma, \gamma)$; the parameters θ and ϵ of the L_{RI} scheme; the performance criterion function $G(\lambda)$ of the Environment.
Output A decision k^+ from the set $D = \{0, \ldots, d\}$. The decision represents the optimal number of "Yes" votes among D.

Method :
Begin
  For i := 1 **To** d **Do**
    $P_0^i := P_1^i := 0.5$
  While $\epsilon < P_0^i < 1 - \epsilon$ **For Any** $i \in \{1, \ldots, d\}$ **Do**
    k := 0
    For i := 1 **To** d **Do**
      y_i := ChooseAction(Δ^i) ; $k := k + y_i$
    EndFor
    For i := 1 **To** d **Do**
      If (y_i = 0) **Then**
        β := GetFeedBack(σ^k)
        If (β = 0) **Then**
          $P_1^i := \theta.P_1^i$; $P_0^i := 1 - P_1^i$
        EndIf
      Else
        β := GetFeedBack(σ^k)
        If (β = 0) **Then**
          $P_0^i := \theta.P_0^i$; $P_1^i := 1 - P_0^i$
        EndIf
      EndIf
    EndFor
  EndWhile
  k^+ := 0
  For i := 1 **To** d **Do**
    If ($P_1^i \geq 1 - \epsilon$) **Then**
      $k^+ := k^+ + 1$
    EndIf
  EndFor
  Return (k^+)
End Procedure ExecuteGooreGame

Fig. 3. Algorithm CGG–AdS: Overall search strategy.