RELATIVE REWARD STRENGTH ALGORITHMS FOR LEARNING AUTOMATA WITH APPLICATIONS TO ROUTING IN COMPUTER NETWORKS

Rahul Simha and James F. Kurose
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

COINS Technical Report 88-05

January 19, 1988

Relative Reward Strength Algorithms for Learning Automata with Applications to Routing in Computer Networks [1]

RAHUL SIMHA & JAMES F. KUROSE

Department of Computer and Information Science
University of Massachusetts
Amherst, Mass. 01003

Abstract

We examine a new class of action probability update algorithms for learning automata that use the *relative* reward strengths of the environment. Specifically, we study update algorithms for S-Model automata in which "recent" environmental responses for each of the actions are retained and used. We prove a convergence result and study the behavior of these automata through simulation. In addition, we describe applications of the automata to routing in computer networks.

[1]This work supported in part by the Office of Naval Research grant N00014-87-K-0304

1. Introduction

Learning automata have been the subject of intense research during the past two decades [4,14,17,26]. Their inherent simplicity and mathematical properties have resulted in wide applicability to various problems, including problems related to computer communication networks [7,20,23]. The environments in which learning automata are designed to operate are typically stochastic in nature, i.e., the environmental response or "reward" to a given automaton action may be considered a random sample drawn from an associated (typically stationary) probability distribution. Broadly speaking, the goal of a learning automaton is to iteratively adjust ("optimize") its action probabilities (defined precisely below) and thus identify that action with the largest expected reward.

Traditionally, learning automata have used only the single, most recent response from the environment in updating the action probabilities [14,20,28] However, it has been noted by several researchers [5,8,25] that the history of interaction with the environment (i.e. previous responses from the environment) may be used in designing more effective action probability update algorithms. These automata have typically used the *entire* past history of the automaton to maintain *estimates* of the mean reward associated with each action. By the Strong Law of Large Numbers, these estimates become increasingly accurate and hence these algorithms converge faster than standard learning automata. However, as pointed out in [25], these automata (at least implicitly) assume a *static* environment and hence are unsuitable for tracking a non-stationary setting. This may be compensated for by the use of partial histories [25] but in this case the speed and provability of convergence may be not be guaranteed.

In this paper, we develop a framework for automata that use such additional information without losing any tracking ability. We present a variation of an S-Model automaton in which the automaton maintains and utilizes the *most recently obtained* reward value for *each* action until the next time that action is selected. A second novel feature of the automata we study is the manner in which these recently obtained reward values are used. Specifically, we study algorithms based on a gradient projection method which updates action probabilities on the basis of *relative* reward strengths. We prove the convergence of one of these algorithms and study its performance through simulation. A major result of the paper is that the performance of this algorithm is superior, in many respects, to that of the well-known SL_{R-I} update algorithm [20,28]. In addition to providing empirical results on the speed of convergence, we investigate several other important issues such as the variance in performance and learning "cost" of update algorithms.

In the next section we define our model of learning automata and environments in the context of standard definitions found in the literature. Section 3 contains a discussion of relative reward strengths and a presentation of the algorithms studied in this paper. In addition, we derive a convergence result for one of the algorithms. In section 4 the application of these learning schemes

to message routing in computer communication networks is discussed. Following that, in section 5 we describe our experimental results. Section 6 contains our conclusions and final remarks.

2. S-Model Learning Automata

In this section we define our model of the S-Model automaton considered in this paper. To place our work in the appropriate context, we present the standard definition (found in [4] and later in [14,17,29]) of a learning automaton and then discuss how the automata studied in this paper may be defined in these terms. A learning automaton is a sextuple $M =< s, \phi, \alpha, P, A, G >$ where

- α is the set of K possible actions that the automaton can take, $\alpha = \{\alpha_i | 1 \leq i \leq K\}$. Where there is no ambiguity we shall refer to action α_i as the i^{th} action.
- ϕ is the set of internal states of the automaton
- P is the probability distribution over the set of states; therefore $P_i^{(n)}$ denotes the probability of selecting state ϕ_i at the n^{th} step.
- G is the output function, $G : \phi \rightarrow \alpha$, that maps the set of internal states to the set of actions. In this paper, as in [17], we take the number of actions and number of states to be the same and G to be a deterministic one-to-one function. In effect, we say that the state and action are one and the same. Thus, $P_i^{(n)}$ denotes the probability of selecting action i at the n^{th} step. In the remainder of the paper we shall refer only to the actions of the automaton, implicitly implying the corresponding internal states.
- s is the set from which the input to the automaton is taken, i.e. the set of possible responses from the environment. If this response is one of two values taken from the set $\{0, 1\}$ then M is a P-Model automaton; if the response is one of a discrete number of values then M is a Q-Model automaton; and finally, if the response is a real number in $[0, 1]$ then M is an S-Model automaton [28]. In this paper, we are concerned with S-Model automata and we denote $s_i^{(n)}$ to be the most recent response for action α_i at step n.
- A is an algorithm or updating scheme that modifies the probability vector $(P_1^{(n)}, ..., P_K^{(n)})$ at each iteration n by using the response from the environment. Typically, this algorithm is executed after each response from the environment. The goal of the algorithm is to hopefully improve the future behavior of the automaton. Typically, in the event that the current environmental reward is high, the algorithm increases the probability of the current action while uniformly decreasing the probability of the other actions [17]. Similarly, if the response implies that the probability of the current action must be decreased, then it is done so while uniformly increasing the probability of the other actions. As will be described later, the

schemes studied in this paper use the relative strengths of the most recently received rewards for each action in order to favor higher reward actions.

We now describe the environment with which the automaton interacts. The environment takes an action (say, action i) as input from the automaton and emits a response $s_i^{(n)}$. We are concerned with models in which the environment responds in a stochastic manner. In the S-Model considered in this paper, for each action i, the environment responds with a reward which is taken from a continuous distribution F_i with mean $E[s_i]$ and thus, each response may be considered to be a *random sample* from the distribution F_i. In this case, the "best" action is that for which $E[s_i]$ is largest and, broadly speaking, the goal of the automaton is to incrementally adjust its action probabilities so that this action is most frequently taken.

3. Relative Reward Strength Algorithms

3.1 Using Old Information

At each step, the automaton can select only one of its actions, to which it will receive exactly one reward value from the environment. Thus, an update mechanism that utilizes reward values for several actions necessarily uses previously-obtained reward values. The notation $s_i^{(n)}$ indicates the most recent response value on action i as of step n. For example, suppose that at the 10^{th} step, the automaton uses the 4^{th} action and obtains the reward 0.6. Next, suppose the 7^{th} action used at the 11^{th} step and a reward of 0.3 is received. Then $s_4^{(10)} = s_4^{(11)} = 0.6$ and $s_7^{(11)} = 0.3$. We note that at each time step n, an action is chosen randomly according to the probability distribution $(P_1^{(n)}, ..., P_K^{(n)})$. If the action chosen at the n^{th} step was the i^{th} action and if the response is denoted by r, then, by our notation, $s_i^{(n)} = r$.

As noted in [25], the standard update algorithms use only the value r in the update algorithm [4,20,28]. A high value of r causes $P_i^{(n+1)}$ to be higher than $P_i^{(n)}$ while decreasing uniformly the probabilities of the other actions. We observe that no information about the *relative* success of the actions taken in the past is taken into account. We are thus motivated to consider the case in which the relative reward strengths of all actions are used in updating the probabilities, i.e., the entire vector s (containing the most recent reward values received as a result of each of the K actions) is used in the update mechanism. Intuitively, we increase action probabilities *in proportion* to the relative size of the rewards last incurred.

The notion of employing the history of environmental responses has been used in [5,24,25]. As previously mentioned, these automata have typically used the *entire* past history of the automaton to maintain *estimates* of the mean (and variance [5]) of the reward distribution associated with each action. These estimates converge almost surely to the mean reward values and hence these

algorithms exihibit faster convergence. However, as noted in [25], these automata (at least implicitly) assume a *static* environment and cannot track a non-stationary environment. The use of a partial past history was addressed in [25], but only briefly and no convergence proofs were obtained for this case.

Relative reward strengths were also studied in [5,24,25]. In [24,25] the difference between the estimated mean rewards for each action was considered in updating the action probabilities. In [5], the relative magnitudes of the estimated mean reward were considered, but as in [24,25], these estimates were obtained over the entire history of the automaton and convergence followed primarily from the convergence of the reward estimates via the Strong Law of Large Numbers. The update algorithms studied in this paper also use the magnitude of the relative reward strengths, but in a different manner than in [5,24,25] (the algorithms in this paper are based on a gradient projection method) and use only a partial history (most recent rewards) as well. Since only the most recent rewards are used as estimates of the relative reward strengths, the algorithms are also well-suited for tracking a non-stationary environment. Extensions of these algorithms to the case in which entire histories (in the form of mean reward estimates) are used are easily made.

3.2 Update Algorithms

At each step, n, the automaton randomly selects an action (according to the current action probabilities, $P_i^{(n)}$), obtains the environment's random response to that action and executes the update algorithm. Thus, only one component of $(s_1^{(n)}, ..., s_n^{(n)})$, is replaced with a new value at each iteration; the other values are those obtained in earlier iterations.

We now present the first gradient-based update mechanism, initially proposed in a microeconomic context [10] and later adopted as the basis for resource allocation algorithms [11,12]. At each iteration n, the change in probability for action i, $\Delta P_i^{(n)}$, computed is proportional to the difference between the reward strength, $s_i^{(n)}$, and the average reward strength over all actions. We now give a precise statement of the update algorithm: $\forall i : P_i^{(n+1)} = P_i^{(n)} + a_n \Delta P_i^{(n)}$, where the change in probability is:

Algorithm $G1$:

$$\begin{aligned} \Delta P_i^{(n)} &= \left(s_i^{(n)} - \frac{1}{|A^{(n)}|} \sum_{j \in A^{(n)}} s_j^{(n)} \right), \qquad \forall i \in A^{(n)} \\ &= 0, \qquad i \notin A^{(n)} \end{aligned} \tag{1}$$

and where a_n is the stepsize parameter at step n and is discussed in detail below. The set $A^{(n)}$ of participating actions is required to ensure that the action probability feasibility constraints are maintained. Typically, $A^{(n)}$ contains all the actions. We note that the set algorithm originally given in [10] is incorrect in certain situations and has since been modified to maintain feasibility [12]. It is computed as follows:

/* Algorithm for computing the set $A^{(n)}$ */

(i) For all i, sort $s_i^{(n)}$.

(ii) Set $A' = \{i |$ action i has largest $s_i^{(n)}\}$.

(iii) Do step (iv) for each $j, j \notin A'$ in descending order of $s_j^{(n)}$.

(iv) If action j would receive a probability $P_j^{(n)} > q_{min}$, as a result of the reallocation defined by equation 1 above with $A^{(n)} = A' \cup \{j\}$, then set $A' = A' \cup j$. The use of q_{min} is discussed below in the section on convergence.

(v) Set $A^{(n)} = A'$

The above update mechanism is based on gradient projection methods [2,12,22] (see [12] for details). In the case of linear constraints, these algorithms have been used to achieve optimal resource allocation in distributed computer systems [9,11,12,27]. They have been demonstrated to possess desirable properties such as *feasibility* (always maintaining a feasible allocation) and *monotonicity* (an increase in the value of the objective function with each iteration, in the case of maximization) in addition to being amenable to decentralization and easy implementation. It is important to note that gradient-based approaches above [2,9,11,12] have been used in deterministic contexts. In this paper, the gradient-based update algorithms use noisy or random estimates (rewards) from the environment in the update process.

We note that similar gradient-based approaches have been used in the analysis and construction of algorithms in connectionist systems [29,30]. The emphasis in [29,30] has been on demonstrating that several established update algorithms [1,14,17] cause the action probability vector to move, on an average, in the direction of the gradient of the performance metric. In contrast, we note that our work is based on a *constrained* optimization formulation. Therefore, the update algorithms we describe here cause the action probability vector to move, on an average, in the direction of the *projection* of the gradient onto the constraint space $\sum_{i=1}^{K} P_i = 1$.

The second algorithm we study is derived directly from the (deterministic) gradient method of [9]. In this mechanism the probability of the action with the highest reward is increased at the expense of reducing the probabilities of *all* the other actions. Thus if action m currently has the highest reward, i.e. $s_m^{(n)} > s_i^{(n)}, \forall i \neq m$, then we may write the change in probability for the other actions (in the feasiblity set $A^{(n)}$) as:

Algorithm $G2$:

$$\Delta P_i^{(n+1)} = \left(s_i^{(n)} - s_m^{(n)}\right) \qquad i \in A^{(n)}, i \neq m \tag{2}$$

The corresponding change for probability P_m is computed as:

$$\Delta P_m^{(n+1)} = -\left(\sum_{i \in A^{(n)}, i \neq m} \Delta P_i^{(n)}\right)$$

The feasibility set $A^{(n)}$ for $G2$ is computed as $A^{(n)} = \{i | P_i^{(n)} + s_i^{(n)} - s_m^{(n)} > q_{min}\}$. We emphasize that the rewards are random samples from a distribution. Thus the algorithms only serve as a guide to improving the probability distribution of the actions. This gives rise to the question of whether the algorithms are in some sense 'optimal'. In the following section we show that under suitable, commonly assumed, restrictions on the stepsizes a_n, algorithm $G1$ converges to the desired optimum vector of action probabilities.

3.3 Convergence

In the above model, the environment responds to each action i with a random sample from distribution F_i with mean $E[s_i]$. We assume that these mean rewards are real numbers in $[0, 1]$. We define the expected reward at step n:

$$E[R_n] = \sum_{i=1}^{K} E[s_i] P_i^{(n)} \tag{3}$$

The objective of the automaton then is to maximize the expected reward by converging to an optimal distribution of action probabilities. We now show that with an appropriate choice of stepsize constants a_n (commonly made by several researchers [5,21]) and some restrictions on the action probabilities, the process with algorithm $G1$ converges in probability to the desired limit.

We restrict the action probabilities such that $q_{min} \leq P_i^{(n)} \leq q_{max}$ where q_{min} and q_{max} are real numbers such that $0 < q_{min} < q_{max} < 1$ and $q_{max} = 1 - (K-1)q_{min}$. This restriction is motivated by the following consideration. Consider the case when the i^{th} action is the optimal action and at the n^{th} iteration, $s_i^{(n)} = 0$ and the action probabilities are now adjusted so that $P_i^{(n+1)} = 0$. In this case $P_i^{(n)}$ remains *absorbed* at zero. It has been noted that previously studied update mechanisms also suffer from this problem (i.e., they converge but not necessarily to correct values) [17,19]. To avoid this case and hence retain the ability to track a non-stationary environment, we restrict the action probabilities to strictly positive values. Stated more precisely, we have the condition

$\forall i : P_i^{(n)} \geq q_{min} > 0$. Since $\sum_{i=1}^{K} P_i^{(n)} = 1$, we immediately obtain the above upper bound on $P_i^{(n)}$. We note here that for our empirical results we took $q_{min} = 0$ and did not encounter absorption to a wrong value.

Clearly, if the i^{th} action is the optimal action, then we would like to have $P_i^{(n)}$ converge to q_{max}, which is the optimum feasible solution to (3). We now prove that under the update algorithm $G1$ the probability of the highest reward action converges to q_{max} in probability, i.e., given arbitrary $\eta, \xi > 0$ there exists N such that $\forall n > N$,

$$Pr[|P_i^{(n)} - q_{max}| > \eta] < \xi$$

THEOREM 1: Let a_n be a sequence of real numbers such that $a_n > 0$, $\sum_{n=1}^{\infty} a_n = \infty$ and $\lim_{n \to \infty} a_n = 0$. Furthermore if the i^{th} action is the optimal action then assume $E[s_i] > E[s_j], j \neq i$. Then $P_i^{(n)}$ converges to q_{max} in probability when algorithm $G1$ is used.

PROOF: (see Appendix)

4. Applications To Routing

In this section we identify possible applications of the above learning algorithms to problems related to message routing in computer networks. Applications of learning automata to routing have been studied in [18] and in the two seminal papers on non-linear environments [20,23]. We only consider linear cost functions and only identify potential applications; a more thorough discussion of this latter consideration can be found in [7,18,20,23].

Consider a computer network in which messages are being transmitted from various sources to different destinations. We consider the case that a source node maintains several possible logical routes for each destination with which it communicates.

4.1 Reliable Routing

Let us consider the transmission of a message across a link that is subject to random transmission errors. We associate with every link a *reliability factor* of message transmission across that link. This might be, for example, the probability that no more than some fixed number of transmissions is ever required for a complete and correct transfer of a message across that link. Given such a reliability factor for each *link* in a route, we associate a reliability factor with the route itself. For example, in the case of the above example, we might construct the reliability factor of a route by taking the product of reliability factors of links that comprise that route. In this paper, we simply assume that there is some technique for estimating the reliability factor of a route with each message sent across it. In practice this estimate might be included in the end-to-end acknowledgment transmitted back

to the source along the same route. In an internet or high-speed network in which only end-to-end error control is performed [3] all reliability information would already be available at a source node.

Consider the situation in which the estimates of the reliability factor are subject to random variations, i.e., that the individual reliability values reported on a message-by-message basis are now random samples from a distribution. In this case we are interested in sending messages over the set of routes in a manner that optimizes the expected reliability factor. Thus, for a particular message, it is desirable to transmit it over a route with the highest mean reliability factor. We now describe how the learning scheme may be used to optimize the transmission of messages. Informally, we make a correspondence between a route and an action, and the reliability estimate of that route when a message is transmitted on it as the environmental response to that action. Then, using the notation established earlier, we define the following:

- α_i - the i^{th} route in the set of K routes, $\{\alpha_1, ..., \alpha_K\}$.
- $P_i^{(n)}$ - the probability that the n^{th} message is transmitted on the i^{th} route.
- $s_i^{(n)}$ - the most recent estimate of the reliability factor of route i after the transmission of the n^{th} message.

Thus, if the i^{th} action is the optimal action then we would like $P_i^{(n)} \rightarrow q_{max}$. If we assume that the mean of the estimator is the mean reliability factor, then by the theorem in the previous section, algorithm $G1$ converges in probability to the optimum solution.

4.2 Minimum Delay Routing

A second application of S-Model automata is in the area of minimum delay routing, where the performance goal is to minimize message transmission delays. Consider the following situation in which a network node must select a route (from a number of logical routes) for sending a message (or series of messages) to a particular destination node. We assume that the message transmission rates of individual nodes are substantially lower than the average traffic rates carried by a single link. This assumption would be appropriate, for example, in a high speed network in which a large number of users can be multiplexed over shared, high bandwidth communication channels. In this case the ***additional*** delay introduced across a link due to traffic from the given node is negligible. The problem thus reduces to finding the logical route for which the delay is the minimum.

Note that the information needed for an automaton to determine the environment's response (a message's delay in this case) is again immediately available at a source node since a copy of every message must be buffered until an end-to-end acknowledgement is received. Now consider the case when the route delay estimates are subject to random variations, i.e., the estimates are random samples from a sampling distribution. The goal is, then, to use the route with the minimum mean

delay. As in the previous subsection, we make a correspondence between routes and actions. We now associate rewards with inverses of the delay estimates, i.e. $s_i^{(n)}$ is the inverse of the most recent estimate of the delay across route i after the transmission of the n^{th} message. If the i^{th} action is the optimal action, then, by Theorem 1, $P_i^{(n)} \rightarrow q_{max}$ in probability.

5. Experimental Results

In this section we present some of our simulation results and demonstrate that algorithm $G1$ produces probability vectors that result in higher performance with less variation than the other algorithms, and hence performs better than the S-Model update algorithm in [28] in several respects.

The experimental framework for these studies was as follows. The automaton had 10 actions, $(\alpha_1, ..., \alpha_{10})$, with mean rewards $(0.8, 0.74, ..., 0.8-0.06i, ..., 0.26)$. The noise in the environment was simulated by taking truncated samples from Gaussian distributions with the above mean values. We present two sets of results: one for the case of high noise (i.e. a high common variance in the Gaussian distributions) and one for the case of zero noise. Since the goal of the automaton is to converge to action probabilities which maximize the expected reward, we are interested in observing the progress the automaton makes with each additional iteration. Consequently, our results are generally in the form of graphs plotting some performance metric as a function of the number of iterations. In our graphs, each point in each curve was obtained as an average of 300 experiments.

We note that there are several ways of selecting stepsizes to satisfy the conditions of Theorem 1, each with varying effects on the performance of the algorithm. We study two such choices for algorithm $G1$: one in which $a_n = 1/n$ and the other where $a_n = a, \forall n \leq N_a$ and $a_n = 1/n, \forall n > N_a$. Stated differently, in the the latter case a constant stepsize is maintained for a short horizon (N_a) before using a decreasing stepsize, whereas in the former choice, the stepsizes are monotonically decreasing. We find that an optimal choice of a constant stepsize for a finite horizon yields better performance than $a_n = 1/n$. The convergence profiles of three algorithms are presented:

- The S-Model update algorithm of [28] with stepsize parameter a (labeled as SL_{R-I}). We note that the stepsize parameter is often referred to as the *learning parameter* in the automata literature [17].
- Algorithm $G1$ with $a_n = 1/n$ (labeled as $G1/n$).
- Algorithm $G1$ with $a_n = a$ and $N_a = 1000$ (labeled as $G1$). The rapidity of performance improvement depends on the choice of a. In figure 1 we plot the expected reward at the 1000^{th} iteration for various stepsizes. From this type of graph, the best stepsizes are selected for the other experiments. For example, in figures 2, 3 and 4 the stepsizes were taken to be 0.04 for algorithm SL_{R-I} and 0.02 for algorithm $G1$.

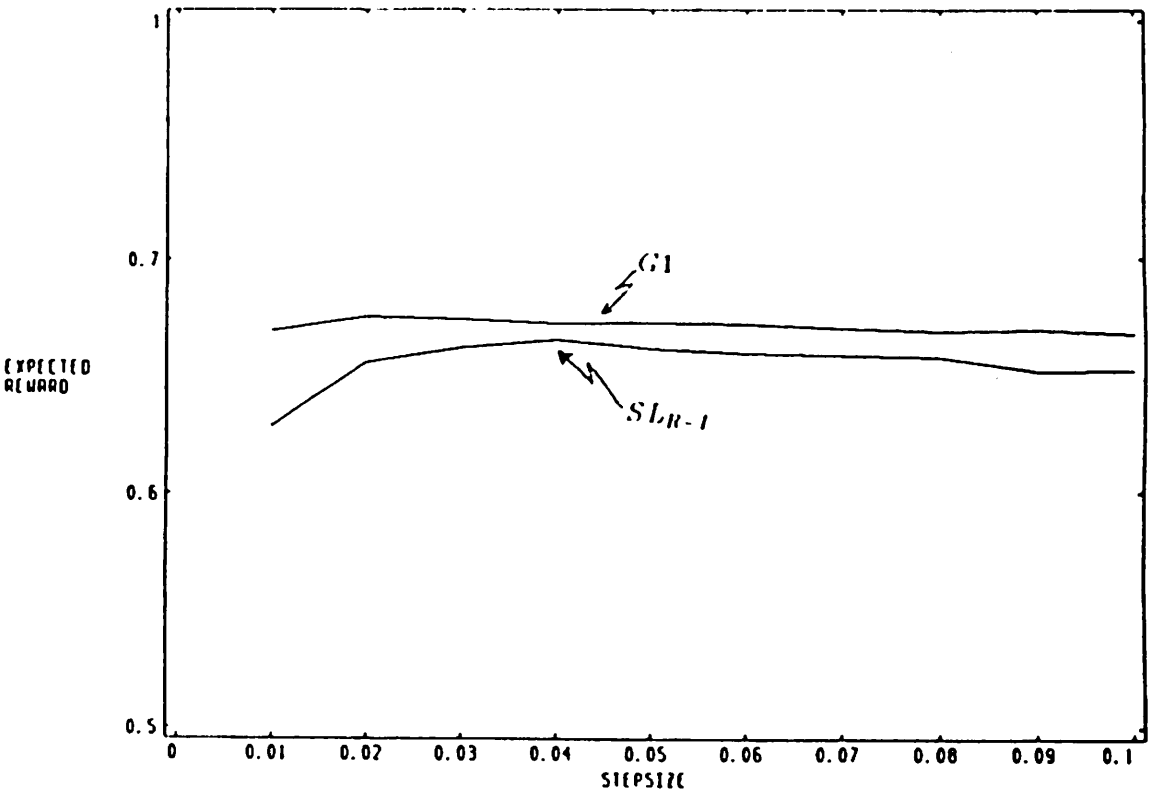

Figure 1: High noise: Expected reward at 1000th iteration vs. stepsize

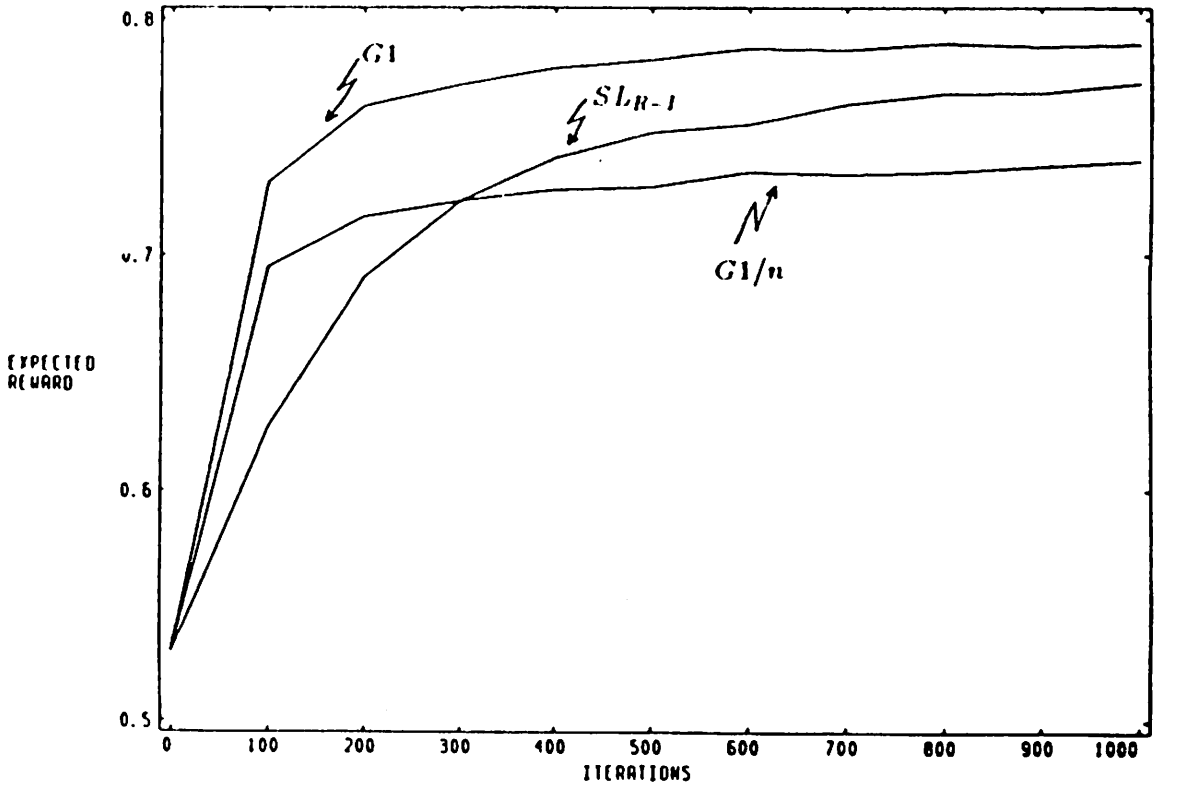

Figure 2: High noise: Expected reward vs. iterations

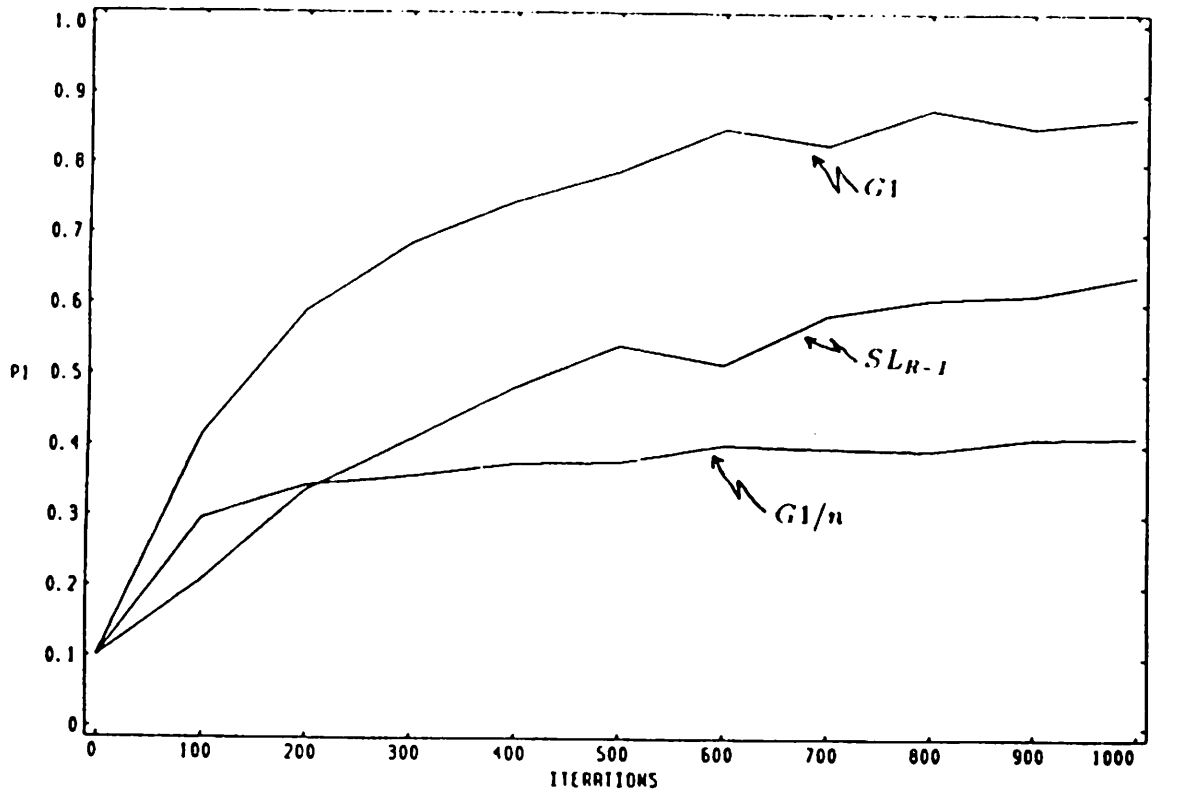

Figure 3: High noise: Probability of highest reward action vs. iterations

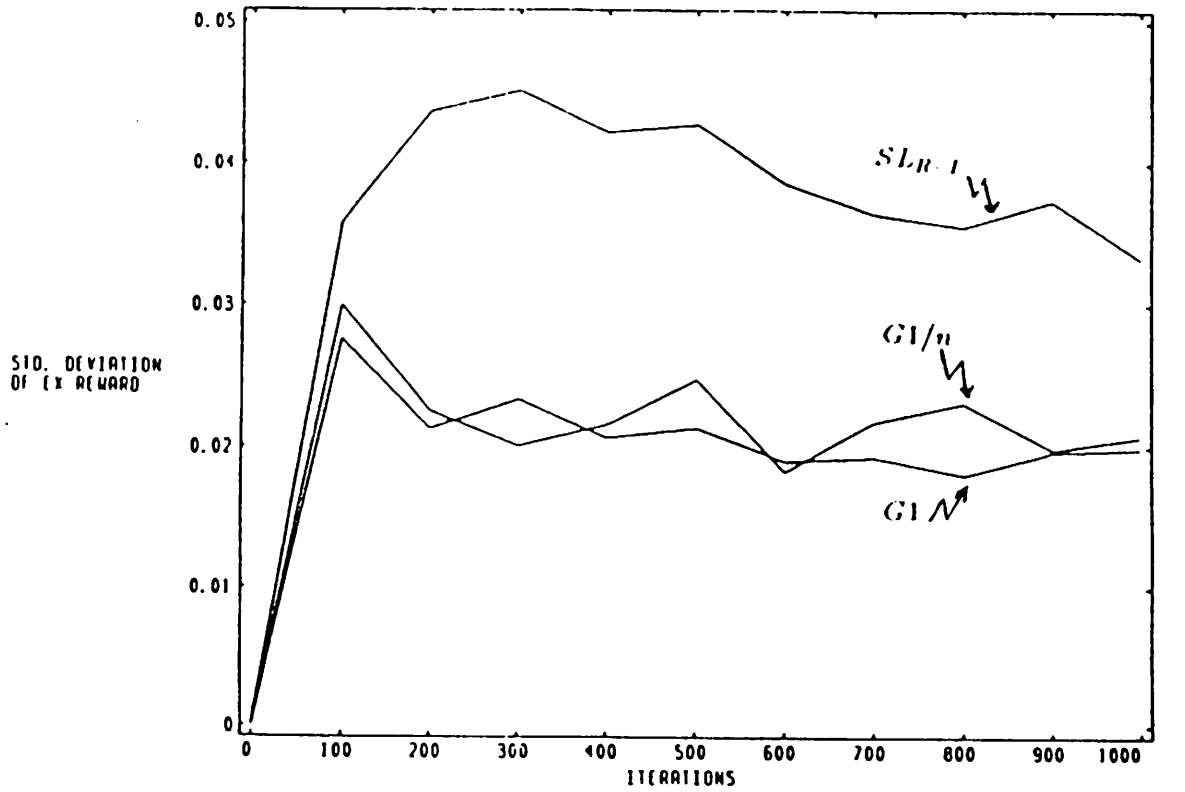

Figure 4: High noise: Standard deviation vs. iterations

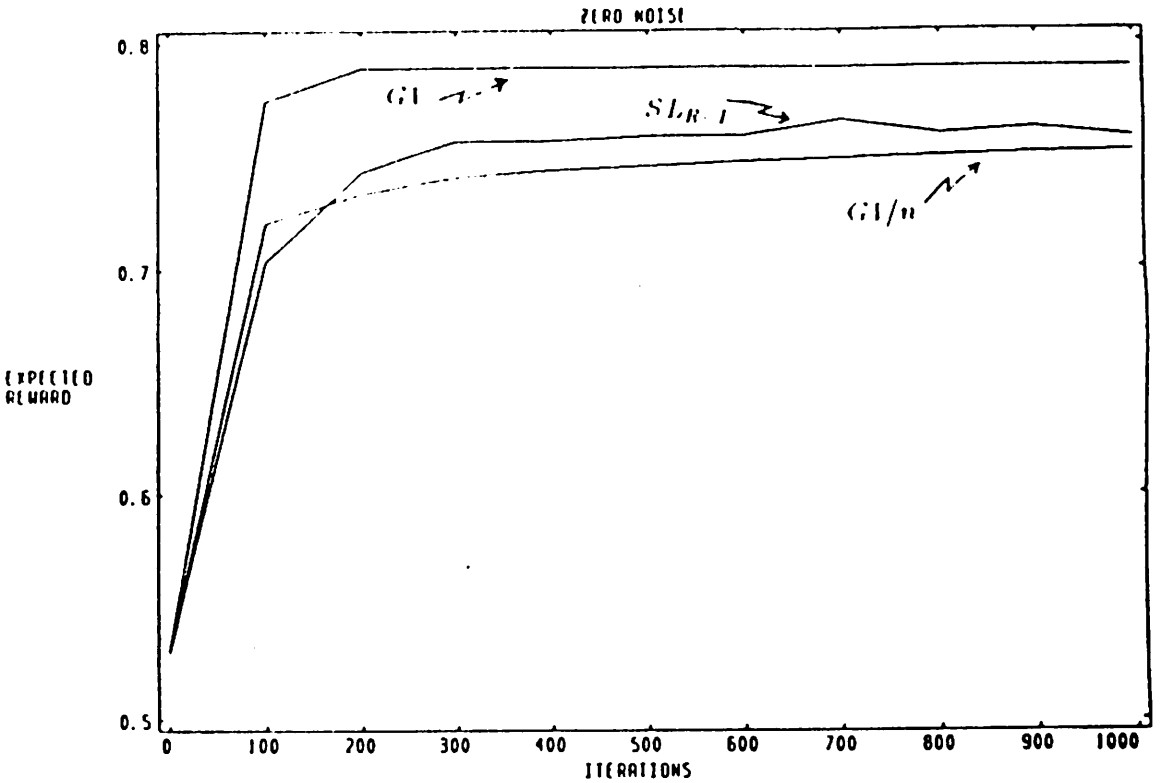

Figure 5: Zero noise: Expected reward vs. iterations

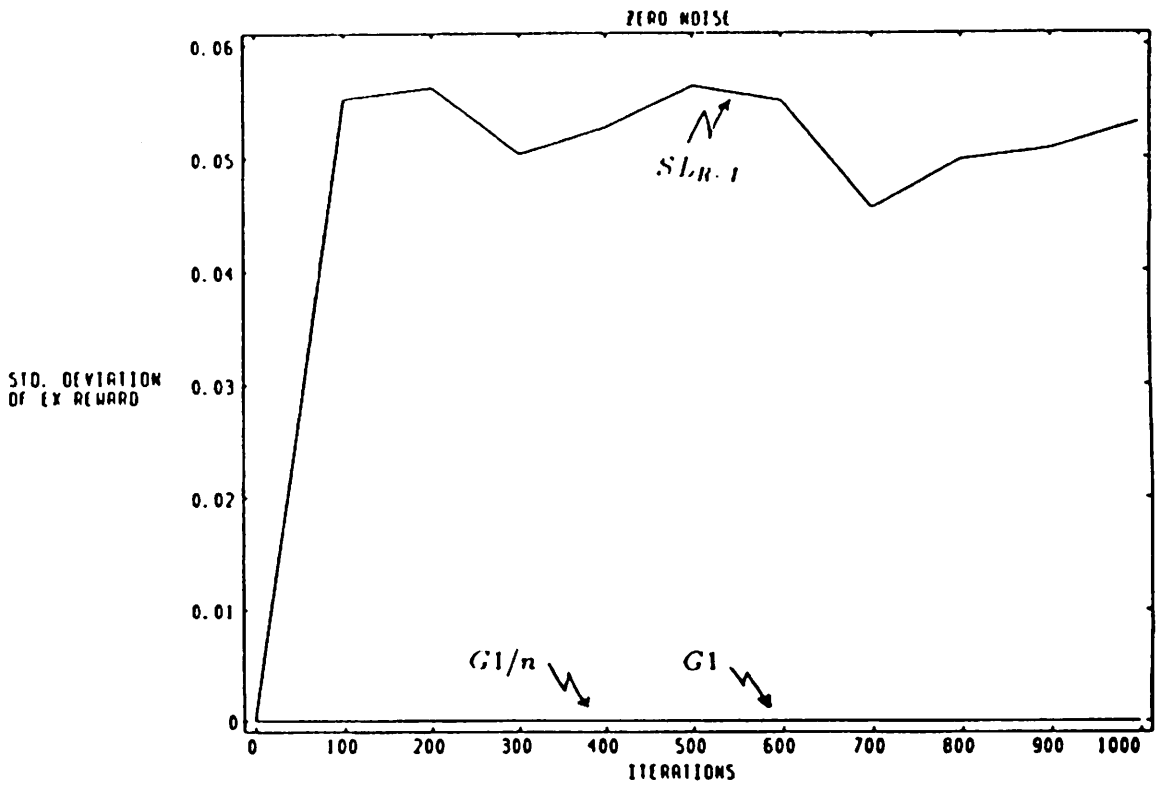

Figure 6: Zero noise: Standard deviation vs. iterations

In figure 2 we plot the expected reward as a function of the number of iterations in the case that the variance of the Gaussian noise process was 0.25. We note that the mean reward produced by algorithm $G1$ is consistently higher than those produced by the other algorithms. It is possible, however, to achieve high expected rewards without converging to the optimal distribution of action probabilities. This might occur, for example, when the relative difference among the expected reward values is small. Thus, it is of interest to observe the trajectory of the probability corresponding to the highest reward action. In figure 3 we plot the probability of the highest reward action (the first action) with increasing iteration number. The trajectory of this probability under algorithm $G1$ is clearly higher than that for the other algorithms. Figure 4 shows the estimated standard deviation at each iteration. Note that the standard deviation curve for algorithm $G1$ lies below the curve for algorithm SL_{R-I}. This indicates that the update mechanism $G1$ not only improves performance sooner but also produces action probability vectors with less variation than does mechanism SL_{R-I}. The curve for algorithm $G1/n$ is approximately the same as that of $G1$.

Next, we study the performance of the algorithms in a situation where the environment responds without noise, i.e., the response for each action is the corresponding mean reward for that action. This approximately indicates the behavior of the automaton in circumstances where the noise content is very low. As can be seen from figure 5, algorithm $G1$ results in higher expected rewards. Figure 6 plots the estimated standard deviation. We note that in this case, algorithms $G1$ and $G1/n$ demonstrate almost no variation in the results they produce, whereas algorithm SL_{R-I} displays a significant amount of variation.

Three additional comments may be made. First, we consider the case when $N_a = \infty$. In this case, algorithm $G1$ displays slight oscillation very close to the optimum whereas algorithm $G1/n$ converges to the optimum. Thus, $G1/n$ will eventually achieve slightly higher expected rewards than $G1$ (with $N_a = \infty$). Our experimental results (not shown here) indicate that this occurs only after an extremely large number of iterations - on the order of $10,000$ to $20,000$ iterations. However, convergence is obtained only in the limit, and thus, for a finite (and more realistic) horizon of, for example, 1000 iterations, a constant stepsize produces better results; for the above experiments we took $N_a = 1000$. Second, we required that for each action, $P_i^{(n)} > q_{min}$ for the proof of convergence. For our experimental results we took $q_{min} = 0$ and did not encounter any cases of absorption. Finally, we note that we have required that the rewards lie in the range $[0, 1]$. In general, for algorithms $G1$ and $G1/n$ this restriction is not necessary. However, for the update mechanism SL_{R-I} [28] the rewards must be scaled to the interval $[0, 1]$.

5.1 Cost incurred during learning

The evaluation of the automaton's performance above is only concerned with convergence to an optimal set of action probabilities. Upon convergence, the actual rewards obtained by the automaton will, in an average sense, be maximized since the expected rewards are maximized.

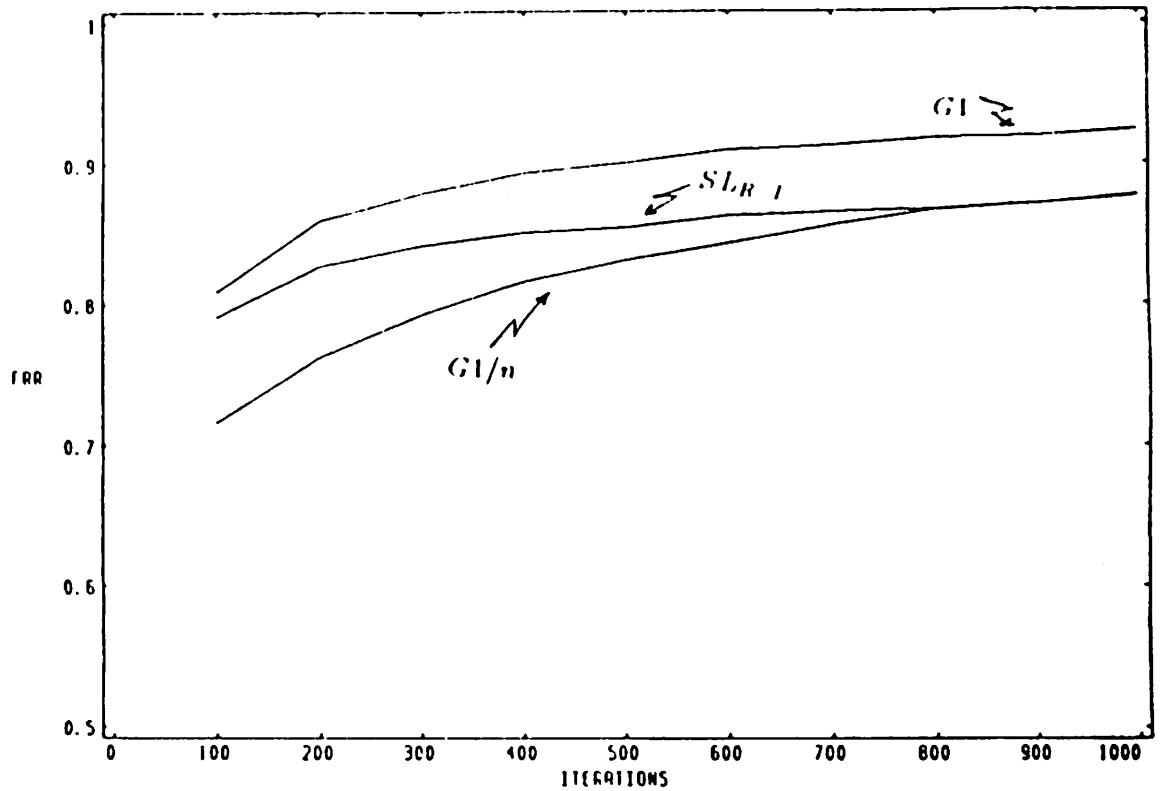

Figure 7: Fractional reward ratio vs. iterations

However, the rewards obtained *during* the learning process are not taken into account in this performance metric. We note that during the learning process, the automaton may frequently sample suboptimal actions (actions whose corresponding expected rewards are not maximal) and thus receive suboptimal rewards. Loosely speaking, an "efficient" algorithm will converge to an optimal set of action probabilities while keeping the cost of learning (sampling suboptimal actions) to a minimum. We are thus motivated to define a performance metric that measures, at each iteration, the cumulative reward obtained since the start of operation of the algorithm. We call this the fractional reward ratio (FRR) and define it as follows. Let the response at iteration i be denoted by r_i. Then if the optimal action is the j^{th} action, FRR_n, the fractional reward ratio at the n^{th} iteration, is defined by

$$FRR_n = \frac{\sum_{i=1}^{n} r_i}{nE[s_j]}$$

Intuitively, if the optimal action, j, were being used since the start of operation then the expected sum total reward at the n^{th} iteration will be $nE[s_j]$. The metric simply computes the ratio between the actual sum total reward obtained and $nE[s_j]$. Thus, if suboptimal actions were taken frequently, then the FRR would be low; if they were taken only occasionally, then the FRR would be high.

Figure 7 shows the FRR plotted against iteration number for each of the three algorithms in the case of high noise. We note that algorithm $G1$ results in a lower learning cost (higher rewards

during the learning phase) as compared to the other algorithms.

6. Conclusions

In this paper we have examined a relative reward approach towards designing update algorithms for S-Model learning automata and suggested their possible applications in routing messages in computer communication networks. One of the algorithms was examined in detail: a convergence result was given and supplemented by experimental results. The performance of this update mechanism was seen to be superior to the SL_{R-I} learning algorithm of [28,20] in many ways. Algorithm $G1$ produced consistently higher performance levels than other algorithms while achieving these levels with lower variance. In addition, the cost of learning was shown to be less than that of other algorithms.

As discussed earlier, the algorithms in this paper are easily modified to use mean reward estimates based on a longer (though not necessarily complete) history of interaction with the environment. In this case it would be important to place these extensions in the context of other such algorithms [5,25]. Learning automata are useful in situations where the characteristics of the underlying system are changing constantly. In that case it is desirable to use algorithms that *track* the changes and maintain a high level of performance. For future work, it would be important to develop a framework to compare the tracking ability of various algorithms.

REFERENCES

[1] A.G.Barto and P.Anandan, "Pattern Recognizing Stochastic Learning Automata," *IEEE Trans. on Man, Systems & Cybernetics*, pp. 360-374, Vol 15, 1985.

[2] D.Bertsekas, E.Gafni and R.Gallager, "Second Derivative Algorithms for Minimum Delay Distributed Routing in Networks," *IEEE Trans. on Communications*, Vol COM-32, No. 12, pp. 911-919, August 1984.

[3] A. Bhargava, J.F. Kurose, D. Towsley and G. Van Leemput, "Performance Comparison of Error Control Schemes in High Speed Computer Communication Networks," *Proc. 1988 IEEE Infocom,* (New Orleans, March, 1988).

[4] B.Chandrasekharan and D.W.C.Shen, "On Expediency and Convergence in Variable Structure Automata," pp. 52-60, *IEEE Trans. on Systems, Man & Cybernetics*, Mar 1968.

[5] L.P.Devroye, "A Class of Optimal Performance Directed Probabilistic Automata," *IEEE Trans. on Systems, Man & Cybernetics*, Vol SMC-6, pp. 777-784, Nov. 1976.

[6] A.Dvoretzky, "On Stochastic Approximation," *Proc. 1959 Berkeley Symp. Math. Stat. and Prob.*, Vol 1, 1959.

[7] Y.M.El Fattah, P.Boyer, A.Dupuis and L.Romoeuf, "Use of a Learning Automaton for Control of Service Activity," Centre National d'Etudes des Telecommunications, Lannion, France.

[8] K.S.Fu and Z.J.Nikolic, "On Some Reinforcement Techniques and their Relation to the Stochastic Approximation," *IEEE Trans. on Automatic Control*, Vol AC-11, No. 4, pp. 756-758, Oct 1966.

[9] R.G.Gallager, "A Minimum Delay Routing Algorithm Using Distributed Computation," pp. 73-85, *IEEE Trans. on Communications*, Vol COM-25, 1977.

[10] G.Heal, "Planning Without Prices," *Rev. of Econ Studies*, Vol 36, pp. 346-362, 1969.

[11] Y.C.Ho, L.Servi and R.Suri, "A Class of Center-Free Resource Allocation Algorithms," *Large Scale Systems*, Vol 1, pp. 51-62, 1980.

[12] J.Kurose and R.Simha, "A Microeconomic Approach to Optimal Resource Allocation in Distributed Computer Systems," *IEEE Trans. on Computers*, to appear.

[13] H.J.Kushner and D.S.Clark, "Stochastic Approximation Methods for Constrained and Unconstrained Systems," *Springer Verlag*, New York, 1978.

[14] S.Lakshmivarahan, "Learning Algorithms: Theory and Applications," *Springer Verlag*, New York.

[15] L.Ljung, "Analysis of Recursive Stochastic Algorithms," *IEEE Trans. on Automatic Control*, Vol AC-20, 1977, pp. 551-575.

[16] L.G.Mason, "An Optimal Learning Algorithm for S-Model Environments," pp. 493-496, *IEEE Trans. on Automatic Control*, Oct 1973.

[17] K.S.Narendra and M.A.L.Thathachar, "Learning Automata - A Survey," pp. 323-334, *IEEE Trans. on Man, Systems & Cybernetics*, July 1974.

[18] K.S.Narendra, E.A.Wright and L.G.Mason, "Applications of Learning Automata to Telephone Traffic Routing Problems," *IEEE Trans. on Systems, Man & Cybernetics*, Vol SMC-7, pp. 785-792, 1977.

[19] K.S.Narendra and S.Lakshmivarahan, "Learning Automata: A Critique," *J. of Cybernetics & Information Science*, Vol 1, 1977.

[20] O.V.Nedzelnitsky,Jr. and K.S.Narendra, "Nonstationary Models of Learning Automata Routing in Data Communication Networks," *IEEE Trans. on Systems, Man & Cybernetics*, Vol SMC-17, No. 6, Nov. 1987.

[21] H.Robbins and S.Munro, "A Stochastic Approximation Method," *Ann. Math. Stat.*, Vol 22, 1951, pp. 400-407

[22] D.Russell, "Optimization Theory," *W.A.Benjamin Inc.*, New York, 1970.

[23] P.R.Srikantakumar and K.S.Narendra, "A Learning Model for Routing in Telephone Networks," *SIAM J. Control & Optimization*, Vol 20, No. 1, Jan 1982.

[24] M.A.L.Thathachar and P.S.Sastry, "A Class of Rapidly Converging Algorithms for Learning Automata," *IEEE Int. Conf. on Cybernetics & Society*, Bombay, India, 1984.

[25] M.A.L.Thatchachar and P.S.Sastry, "A New Approach to the Design of Reinforcement Schemes for Learning Automata," *IEEE Trans. on Systems, Man & Cybernetics*, Vol SMC-15, No. 1, Jan. 1985.

[26] M.L.Tsetlin, "On the Behavior of Finite Automata in Random Media," *Automat. i. Telemekh*, Vol. 22, 1961.

[27] J.N.Tsitsiklis and D.P.Bertsekas, "Distributed Asynchronous Optimal Routing in Data Networks," *Proc. of the 23rd Conf. on Decision & Control*, 1984.

[28] R.Viswanathan and K.S.Narendra, "Stochastic Automata Models with Applications to Learning Systems," *IEEE Trans. on Systems, Man & Cybernetics*, Jan 1973.

[29] R.J.Williams, "Reinforcement Learning in Connectionist Networks: A Mathematical Analysis," Report 8605, *Institute for Cognitive Science*, University of California, San Diego, 1986.

[30] R.J.Williams, "Reinforcement-Learning Connectionist Systems," *Technical Report NU-CCS-87-3*, College of Computer Science, Northeastern University, Boston, 1987.

[31] J.Wolfowitz, "On the Stochastic Approximation Method of Robbins and Munro," *Ann. Math. Stat.*, Vol 25 pp. 457-461, 1952.

7. Appendix

Our proof techniques closely follow that of [21] and we use a result due to [31]. We are at present uncertain about whether the convergence of our algorithms is a consequence of any of the general theorems of [6][13][15] and instead, we prove our results directly. We believe that due to the *combinatorial* nature of the the computation of the feasibility set $A^{(n)}$ (it includes a *sort* operation), these theorems are not applicable here. In fact, we note that while most learning algorithms [25][14] have some martingale property, the algorithms presented here can be shown not to possess any such property. In addition, we note that while our proof has been presented for the case of the S-Model,

the *same* proof is valid for P-Model extensions of the relative reward algorithms. In order to simplify the presentation of our proof we define the following terms:

1. Let I_B denote the interval $[q_{min}, q_{max} - \frac{\epsilon}{2}]$.
2. Let I_C denote the interval $[q_{max} - \frac{\epsilon}{2}, q_{max}]$.
3. Assume without loss of generality that the first action is the highest reward action i.e., that $E[s_1] > E[s_i], i \neq 1$. We will show that $P_1^{(n)}$ converges to q_{max} in probability.
4. Define a distance to the I_C interval as follows:

$$\begin{aligned} L(P_1^{(n)}) &= 0 \quad if \quad P_1^{(n)} \in I_C \\ &= q_{max} - \frac{\epsilon}{2} - P_1^{(n)} \quad if \quad P_1^{(n)} \in I_B \end{aligned}$$

5. Let:

$$\begin{aligned} P_B^{(n)} &= Pr[P_1^{(n)} \in I_B] \\ P_C^{(n)} &= Pr[P_1^{(n)} \in I_C] \\ b_n &= E[L^2(P_1^{(n)})] \\ d_n &= E[(q_{max} - \frac{\epsilon}{2} - P_1^{(n)})\Delta P_1^{(n)} | P_1^{(n)} \in I_B] \\ e_n &= E[\Delta^2 P_1^{(n)} | P_1^{(n)} \in I_B] \\ f_n &= E[L^2(P_1^{(n+1)}) | P_1^{(n)} \in I_C] \end{aligned}$$

We first prove a set of lemmas and later use these lemmas in constructing our proof.

LEMMA 1: $\exists M : \forall n > M, E[\Delta P_1^{(n)} | P_1^{(n)} \in I_B] > 0$

PROOF: Define $[1 \in A^{(n)}]$ as the event that node 1 belongs to the set $A^{(n)}$ (section 3.3). We have

$$\begin{aligned} E[\Delta P_1^{(n)} | P_1^{(n)} \in I_B] &= E[\Delta P_1^{(n)} | P_1^{(n)} \in I_B \wedge 1 \in A^{(n)}] Pr[1 \in A^{(n)}] \\ &\quad + E[\Delta P_1^{(n)} | P_1^{(n)} \in I_B \wedge 1 \notin A^{(n)}] Pr[1 \notin A^{(n)}] \\ &= E[\Delta P_1^{(n)} | P_1^{(n)} \in I_B \wedge 1 \in A^{(n)}] Pr[1 \in A^{(n)}] \qquad (4) \end{aligned}$$

We now establish that each term of the product in equation 4 is strictly positive.

Since each $s_i^{(n)}$ in equation 1 is bounded, $\left|\Delta P_i^{(n)}\right|$ is bounded. Denote this bound by ΔP_{max}. Since $a_n \rightarrow 0$, we can choose M such that for all $n > M$: $a_n \Delta P_{max} \leq \frac{\epsilon}{2(K-1)}$. Consider

$n > M$. Since $P_1^{(n)} \in I_B$, $P_1^{(n)} \leq q_{max} - \frac{\epsilon}{2}$ and thus there must exist at least one action $j \neq 1$, such that $P_j^{(n)} \geq \frac{\epsilon}{2(K-1)}$ in order to satisfy $\sum_{i=1}^{K} P_i^{(n)} = 1$. Thus the cardinality of $A^{(n)}$ must be at least two. Expanding the first term in the product in equation 4:

$$\begin{aligned} E[\Delta P_1^{(n)} | P_1^{(n)} \in I_B \wedge 1 \in A^{(n)}] &= E[s_1^{(n)} - \frac{1}{|A^{(n)}|} \sum_{j \in A^{(n)}} [s_j^{(n)}] \\ &= E[s_1^{(n)}] - \frac{1}{|A^{(n)}|} \sum_{j \in A^{(n)}} E[s_j^{(n)}] \\ &> 0 \end{aligned}$$

where the inequality follows directly from the cardinality of $A^{(n)}$, the fact that $E[s_i^{(n)}] = E[s_i]$, and the assumption that $E[s_1] > E[s_j] \ \forall j \neq 1$.

We next establish that $Pr[1 \in A^{(n)}]$ is strictly positive. Note that the event $[s_1^{(n)} > s_j^{(n)}, \forall j \neq 1]$ implies the event $[1 \in A^{(n)}]$. Consider now the vector of rewards $(s_1^{(l)}, s_2^{(l)}, ..., s_K^{(l)})$ for any step l. Since for each action i, $P_i^{(n)} \geq q_{min}$, there is a non-zero probability ($\geq q_{min}^K$) that each of the K actions will have been sampled within K steps. Then, since $E[s_1] > E[s_j], j \neq 1$, we have $Pr[s_1^{(l+K)} > s_j^{(l+K)} \forall j \neq 1] > 0 \forall l$ and hence $Pr[1 \in A^{(n)}]$ is non-zero. □.

LEMMA 2: b_n satisfies the following recurrence relation:

$$b_{n+1} = f_n P_C^{(n)} + b_n + P_B^{(n)}(a_n^2 e_n + g_n - 2a_n d_n)$$

where g_n is a term bounded as follows: $|g_n| \leq a_n^2 \Delta^2 P_{max}^{(n)}$.

PROOF: We have

$$E[L^2(P_1^{(n)})] = E[L^2(P_1^{(n)}) | P_1^{(n)} \in I_C] Pr[P_1^{(n)} \in I_C] + E[L^2(P_1^{(n)}) | P_1^{(n)} \in I_B] Pr[P_1^{(n)} \in I_B]$$

Thus,

$$E[L^2(P_1^{(n)})] = E[L^2(P_1^{(n)}) | P_1^{(n)} \in I_B] Pr[P_1^{(n)} \in I_B] \tag{5}$$

since by definition $[L^2(P_1^{(n)}) | P_1^{(n)} \in I_C] = 0$. Next we expand $E[L^2(P_1^{(n+1)})]$ in a similar manner:

$$\begin{aligned} E[L^2(P_1^{(n+1)})] &= E[L^2(P_1^{(n+1)}) | P_1^{(n)} \in I_C] Pr[P_1^{(n)} \in I_C] \\ &+ E[L^2(P_1^{(n+1)}) | P_1^{(n)} \in I_B] Pr[P_1^{(n)} \in I_B] \end{aligned} \tag{6}$$

The first part of the second term on the right can be written as

$$E[L^2(P_1^{(n+1)}) | P_1^{(n)} \in I_B] = E[(q_{max} - \frac{\epsilon}{2} - P_1^{(n)} - a_n \Delta P_1^{(n)})^2 | P_1^{(n)} \in I_B] + g_n \tag{7}$$

Since the difference between $P_1^{(n+1)}$ and $P_1^{(n)}$ is at most $a_n \Delta P_{max}^{(n)}$, we have: $|g_n| \leq a_n^2 \Delta^2 P_{max}^{(n)}$. We now expand the first term on the right hand side of equation (7) as:

$$\begin{aligned} E[(q_{max} - \frac{\epsilon}{2} - P_1^{(n)} - a_n \Delta P_1^{(n)})^2 | P_1^{(n)} \in I_B] &= E[(q_{max} - \frac{\epsilon}{2} - P_1^{(n)})^2 | P_1^{(n)} \in I_B] \\ &\quad -2a_n E[(q_{max} - \frac{\epsilon}{2} - P_1^{(n)}) \Delta P_1^{(n)} | P_1^{(n)} \in I_B] \\ &\quad +a_n^2 E[\Delta^2 P_1^{(n)} | P_1^{(n)} \in I_B] \end{aligned} \tag{8}$$

Combining equations (5),(7) and (8) we get

$$E[L^2(P_1^{(n+1)}) | P_1^{(n)} \in I_B] Pr[P_1^{(n)} \in I_B] = b_n + P_B^{(n)}(a_n^2 e_n + g_n - 2a_n d_n) \tag{9}$$

Substituting (9) in (6) we get

$$b_{n+1} = f_n P_C^{(n)} + b_n + P_B^{(n)}(a_n^2 e_n + g_n - 2a_n d_n) \tag{10}$$

which is the required recurrence relation □.

PROOF(of Theorem 1): We expand the recurrence relation (10) of lemma 2 and rearrange the terms to get:

$$2\sum_{i=1}^{n} P_B^{(i)} a_i d_i = b_1 - b_{n+1} + \sum_{i=1}^{n} f_i P_C^{(i)} + \sum_{i=1}^{n} P_B^{(i)}(a_i^2 + g_i)$$

Noting that $\forall n\ b_n \geq 0$, and taking the summations to infinity gives:

$$2\sum_{i=1}^{\infty} P_B^{(i)} a_i d_i \leq b_1 + \sum_{i=1}^{\infty} f_i P_C^{(i)} + \sum_{i=1}^{\infty} P_B^{(i)}(a_i^2 e_i + g_i) \tag{11}$$

Since each of the terms $a_i^2 e_i$, f_i and g_i is bounded above by $a_i^2 \Delta^2 P_{max}^{(n)}$, the two series on the right of inequality 11 converge. Thus the series on the left, $\sum_{i=1}^{\infty} P_B^{(n)} a_i d_i$ is bounded above. Now, choosing M to satisfy the condition of lemma 1, we have from lemma 1 that $a_n d_n \geq 0$ and thus $\sum_{i=M}^{\infty} P_B^{(n)} a_i d_i$ is bounded and increasing and, therefore, $\sum_{i=1}^{\infty} P_B^{(n)} a_i d_i$ exists and is finite.

We now consider two cases. First, suppose the series $\sum_{i=1}^{\infty} P_B^{(n)} a_i d_i$ exists because the sequence $P_B^{(n)}$ converges to zero. In this case, Theorem 1 is proved directly, since $P_B^{(n)} \rightarrow 0$ implies that for $\epsilon/2 > 0$ and $\eta > 0$ there exists N such that $\forall n > N$, $Pr[q_{max} - \frac{\epsilon}{2} - P_1^{(n)} > \epsilon/2] < \eta$ or $Pr[q_{max} - P_1^{(n)} > \epsilon] < \eta$.

In the second case, assume that $P_B^{(n)}$ does not converge to zero. Then there exists an infinite subsequence $P_B^{(n_1)}, P_B^{(n_2)}$... such that $\inf_i P_B^{(n_i)} > 0$. We now consider two further cases, one

in which $\exists M' : \forall n > M' \ \inf_{n \geq M'} P_B^{(n)} > 0$ and the other in which there does not exist an M' that satisfies the above condition.

CASE 1: $\exists M' : \forall n > M' \ \inf_{n \geq M'} P_B^{(n)} > 0$. We then get

$$\sum_{i=1}^{M'} P_B^{(i)} a_i d_i + \inf_{i \geq M'} P_B^{(i)} \sum_{i=M'}^{\infty} a_i d_i \leq \sum_{i=1}^{\infty} P_B^{(i)} a_i d_i < \infty.$$

and thus $\sum_{i=1}^{\infty} a_i d_i$ converges. We note that under the conditions of Theorem 1, $\sum_{i=1}^{\infty} a_i = \infty$ and therefore $\liminf_{n \to \infty} d_n = 0$. We have shown that statement (12) in [31] holds true and following the rest of the proof in [31], we conclude that $P_1^{(n)}$ converges to q_{max} in probability.

CASE 2: $\nexists M' : \forall n > M' \ \inf_{n \geq M'} P_B^{(n)} > 0$. Then $\exists \xi > 0$ such that for some sample tracks (with strictly positive probability) there is an infinite subsequence $P_1^{(n_1)}, P_1^{(n_2)}, \ldots$ where each $P_1^{(n_i)} < q_{max} - \frac{\epsilon}{2} - \xi$. For, if there were no $\xi > 0$ that satisfied the above condition, then that would imply that $P_B^{(n)}$ converges to zero. Thus there exists a set of sample paths that occurs with strictly positive probability such that on each sample path $P_1^{(n)}$ makes an infinite number of crossings between $q_{max} - \frac{\epsilon}{2}$ and $q_{max} - \frac{\epsilon}{2} - \xi$. Let $\gamma(i)$ and $\delta(i)$ denote the stopping times when $P_1^{(\gamma(i))} > q_{max} - \frac{\epsilon}{2}$ and $P_1^{(\delta(i))} < q_{max} - \frac{\epsilon}{2} - \xi$. Since $a_n \to 0$, $\lim_{n \to \infty}(\delta(n) - \gamma(n)) = \infty$. Hence the sums $\sum_{k=\gamma(n)}^{\delta(n)} \Delta P_1^k$ must get arbitrarily small (arbitrarily close to $-\infty$). But for $P_1^{(n)} \in I_B$, $E[\Delta P_1(n)] > 0$, and so, by the Strong Law of Large Numbers, $\sum_{i=1}^{n} E[\Delta P_1(i) | P_1^{(i)} \in I_B] \to \infty$. This contradicts the claim that $\sum_{k=\gamma(n)}^{\delta(n)} \Delta P_1^k$ gets arbitrarily small. Therefore, we get our result that $P_1^{(n)}$ converges to q_{max} in probability $\square$.