Multiple Estimation Models for Faster Reinforcement Learning

Kumpati S. Narendra†, Yu Wang‡, Snehasis Mukhopadhay*, and Nicholas Nordlund†

Center for Systems Science, Yale University

Abstract—In a recent paper the authors proposed a new approach to reinforcement learning based on multiple estimation models. Simple situations involving the use of direct schemes in learning automata, and indirect (estimation based) schemes in feed-forward networks, were presented. The simulation results demonstrated that the proposed schemes are an order of magnitude faster than the linear reward-inaction scheme of learning automata, and comparable to the indirect scheme based on the pursuit algorithm. At the same time they are also substantially more robust than the latter. This makes them attractive in practical applications which are significantly more complex due to interacting decision makers.

The main objective of this paper is two fold:
(i) To provide reasons for the observed robustness of the proposed scheme and
(ii) To demonstrate through simulation studies that the scheme performs even better than the pursuit algorithm in more complex situations involving feedback (i.e. Markov Decision Processes MDP). A simulation study of Blackjack is also included in the last section.

I. INTRODUCTION

Originally based on mathematical psychology models of animal and child learning, reinforcement learning aims to find optimal decisions (decision rules) in unknown or uncertain environments using a qualitative and noisy on-line performance feedback. At the present time, reinforcement learning is one of the most active areas of research in systems theory. A large variety of learning models and algorithms ([1], [2]) have been proposed in the past fifty years. One of the earliest learning schemes proposed is the learning automaton which was investigated extensively in the 1960s. The aim of such automata was to determine “the best” of a finite set of actions acting in a stationary random environment based on the rewards obtained by the actions. Since each action is assumed to have a different probability of reward, this corresponded to determining the maximum probability of a reward. In more general cases where the actual reward is quantified, the objective is to determine the action with the highest expected reward. The numerous algorithms which were developed along with their convergence properties are collected together in a book by Narendra and Tharthachar [3].

In course of time the above algorithms were extended to multiple states and decision makers, dynamic environments and finite number of states (Markov Decision Processes) and to the continuous time case using stochastic differential equations. As noted by Sutton and Barto [2] in their recent book, all the methods and algorithms trace their origins back to trial and error used in the learning automaton, and optimal control methods initiated by Pontryagin [4] and his co-workers and the methods of Hamilton-Jacobi, and more recently Bellman [5]. In recent years methods broadly termed as Approximate Dynamic Programming (also referred to as Neurodynamic Programming) have been proposed for approximating the optimal policy. It is clear from the above comments that all recent methods combine trial and error search with optimal control methods and that the unifying theme of all algorithms is essentially a probabilistic search over an action/decision probability space.

Model free(Direct) and Model-based (Indirect) Methods: The large class of algorithms that are currently known can be broadly classified into two classes: (i) Model Free (ii) Model Based. The learning automaton belongs to the first class in which the aim is to directly learn the optimal policy, without attempting to learn about the environment. Model-based methods such as Adaptive Dynamic Programming and Approximate Dynamic Programming, on the other hand maintain and update a model of the environment and use it, in turn, to compute the optimal policy.

In this paper we will be interested in the behavior of the learning automaton—(a model free method) and two model based methods. The first of these is Pursuit Learning-(PL) which has proposed in 1985, and the multiple model based method (MM), which is the main topic of this paper.

In [6], the authors proposed the multiple models as a new method of optimizing in random environments. Some simple learning situations were considered to indicate the improvement in performance. The principal objective of this paper is to describe the method in considerably greater detail, discuss its convergence properties, and illustrate its advantages by applying it to substantially more complex systems.

II. THE LEARNING AUTOMATON, THE PURSUIT ALGORITHM (PL) AND THE MULTIPLE MODELS APPROACH (MM)

Since we will be using concepts related to the learning automaton and since our main comparison of the new approach is with the indirect pursuit algorithm, we briefly describe the main features of all of them.
If at stage $n$, an action $\alpha(n)$ can be performed and probabilistically yields a reward $\beta(i) = 1$ or a penalty $\beta(i) = 0$. The environment is defined by $r$ reward probabilities $d_1, d_2, \cdots, d_r$, where $d_i = \text{Prob}(\beta = 1 | \alpha = \alpha_i)$. The maximum of the probabilities is $d_{\text{opt}} = \arg \max_i d_i$ and corresponds to the action $\alpha_{\text{opt}}$. The principal question to be answered here is what strategy should an automaton use to update the probabilities of its actions (i.e. $p_i$) at state $(n+1)$ on the basis of a reward or a penalty response to an input at stage $n$? If the process (starting with uniform probabilities $1/n$) continues, how do $p_i(n)$ evolve? do they converge? what is their asymptotic behavior? and is the reward increasing at every step in an expected sense? are some of the questions of interest, and these have been investigated in detail in the literature [6]. It is now known that if probabilities are increased or decreased with a fixed gain $\eta$ in the learning algorithms, the best that can be expected is $\epsilon$-optimality (to be defined later). For the system to be strictly optimal, the gain parameter $\eta$ has to be decreased asymptotically to zero. In this case $\alpha_{\text{opt}}$ will be chosen in the limit w.p. 1. However, since learning schemes are notoriously slow, our interest is only in fixed values of $\eta$ and hence $\epsilon$-optimality. It is also common knowledge that the larger the value of $\eta$, the faster is the convergence rate, but greater is the probability of choosing a non-optimal action in the limit.

We conclude this section on the learning automaton with the Linear-Reward Inaction Scheme [3], (which is one of a very large number of such schemes), which is $\epsilon$-optimal, and used in the following sections for comparison purposes.

The $L_{R-I}$ scheme: If at stage $n$, with probabilities of actions equal to $p_i(n)$ ($i = \{1, \cdots, r\}$), $\alpha_i$ is chosen, the updating of action probabilities proceeds as follows:

$$p_i(n+1) = p_i(n) + \frac{1}{\eta} g[p_i(n)] \quad (1)$$

where $g[p_i(n)]$ in the $L_{R-I}$ scheme is $\eta p_j(n)$, so that

$$\sum_{j \neq i} g[p_j(n)] = \eta [1 - p_i(n)]$$

A. The Learning Automaton:

A random environment $E$ with an input set $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_r\}$ and an output set $\beta = \{0, 1\}$ is shown in Figure (1).

At stage $n$, an action $\alpha(n)$ can be performed and probabilistically yields a reward $\beta(i) = 1$ or a penalty $\beta(i) = 0$. The environment is defined by $r$ reward probabilities $d_1, d_2, \cdots, d_r$, where $d_i = \text{Prob}(\beta = 1 | \alpha = \alpha_i)$. The maximum of the probabilities is $d_{\text{opt}} = \arg \max_i d_i$ and corresponds to the action $\alpha_{\text{opt}}$. The principal question to be answered here is what strategy should an automaton use to update the probabilities of its actions (i.e. $p_i(n)$) at state $(n+1)$ on the basis of a reward or a penalty response to an input at stage $n$? If the process (starting with uniform probabilities $1/n$) continues, how do $p_i(n)$ evolve? do they converge? what is their asymptotic behavior? and is the reward increasing at every step in an expected sense? are some of the questions of interest, and these have been investigated in detail in the literature [6]. It is now known that if probabilities are increased or decreased with a fixed gain $\eta$ in the learning algorithms, the best that can be expected is $\epsilon$-optimality (to be defined later). For the system to be strictly optimal, the gain parameter $\eta$ has to be decreased asymptotically to zero. In this case $\alpha_{\text{opt}}$ will be chosen in the limit w.p. 1. However, since learning schemes are notoriously slow, our interest is only in fixed values of $\eta$ and hence $\epsilon$-optimality. It is also common knowledge that the larger the value of $\eta$, the faster is the convergence rate, but greater is the probability of choosing a non-optimal action in the limit.

We conclude this section on the learning automaton with the Linear-Reward Inaction Scheme [3], (which is one of a very large number of such schemes), which is $\epsilon$-optimal, and used in the following sections for comparison purposes.

The $L_{R-I}$ scheme: If at stage $n$, with probabilities of actions equal to $p_i(n)$ ($i = \{1, \cdots, r\}$), $\alpha_i$ is chosen, the updating of action probabilities proceeds as follows:

$$p_i(n+1) = p_i(n) + \frac{1}{\eta} g[p_i(n)] \quad \beta(n) = 1$$

$$p_j(n+1) = p_j(n) - g[p_j(n)] \quad \beta(n) = 0$$

$g[p_j(n)]$ in the $L_{R-I}$ scheme is $\eta p_j(n)$, so that

$$\sum_{j \neq i} g[p_j(n)] = \eta [1 - p_i(n)]$$

B. The Pursuit Algorithm

The learning automaton considered earlier is a direct scheme, since it uses only the environmental feedback at every iteration. As stated earlier, indirect algorithms use the entire history of the environmental feedback, to order the preference of the actions at every stage $n$ [7]. A special case of such algorithms is the pursuit algorithm to converge faster and more accurately than the $L_{R-I}$ scheme treated earlier, and this has been extensively verified in the literature. It is this algorithm that we use as a benchmark for comparison purposes, and hence we give further details concerning the method.

Let $n = \sum_{i=1}^{T} N_i$, where $N_i$ is the number of times the action $\alpha_i$ has been chosen. Let $n_i$ be the number of rewards obtained. The decision maker maintains the estimates $\hat{d}_i = \frac{n_i}{N_i}$ for all the actions and considers $\arg \max_i d_i$ as the optimal action at that stage. It is the probability of this action which is increased at every stage while the probabilities of all the other actions are decreased (as described for the $L_{R-I}$ scheme).

Assuming that at stage $n$, a non-optimal action $\alpha_j$ is chosen, the following three cases need to be considered:

1) If $\alpha_j$ results in a reward: $\hat{d}_j$ is updated as $\hat{d}_j = \frac{n_j + 1}{N_j + 1}$

2a) $\frac{n_j + 1}{N_j + 1} > \frac{n_i}{N_i}, \hat{d}_j$ is the optimal estimate, and if

2b) $\frac{n_j + 1}{N_j + 1} < \frac{n_i}{N_i}, \hat{d}_i$ is the optimal estimate

2) If $\alpha_j$ results in a penalty, then $\frac{n_i}{N_i} < \frac{n_j}{N_j} < \frac{n_i}{N_i}$, and hence $\hat{d}_i$ is the optimal estimate.

In the first case, the probability $p_j(n)$ is increased, and in the latter two cases 1b) and 2, it is the probability of action $\alpha_i$ that is increased.

The convergence properties of this scheme are treated briefly in the next section.

C. The Multiple Models Approach

The new approach suggested in [6] for use in learning schemes involves multiple models or multiple estimators. Such estimates are associated with each action in the entire system. For the sake of simplicity, the principle is explained in terms of the actions of a learning automaton (as shown in section IV, the method can be used in Markov Decision Processes and also in games that are complex, such as Blackjack in section V).

The simple objective in the automaton is to determine the estimates of the reward probabilities, and to increase the one that corresponds to the maximum estimate. This procedure, in turn has to be shown to have desirable convergence properties, and this is considered in section IV. In this section, we merely outline the procedure for estimating $d_i$.

Comment 2: Both adaptive and fixed estimators have been considered, but in this paper we confine our attention to fixed estimates.

Corresponding to action $\alpha_i$ are $m$ fixed probabilities chosen uniformly in the interval $[0, 1]$. For purposes of
The number of estimators was arbitrarily chosen to be 9 with fixed probabilities 0.1, 0.2, · · · , 0.9. The aim is to choose one of these as the best estimate of the reward probabilities, on the basis of the responses observed in the system.

Let the action \( \alpha_i \) be chosen \( N_i \) times and yield a reward \( n_i \) times (note that \( \frac{n_i}{N_i} \) is the estimate used in the pursuit algorithm).

To determine which of the constant probabilities best explains the observed random sequence, we use the likelihood function:

\[
f[m_j, N_i] = m_j^{n_i}(1 - m_j)^{N_i - n_i}
\]

where \( n_i \) represents the number of rewards and \( N_i \) the number of times the action is tried. It is seen that the real number \( m \) that maximizes the function is \( \frac{n_i}{N_i} \), which corresponds to that used in pursuit learning.

In the MM approach, the value of \( m_j \in (0.1, \cdots, 0.9) \) which is maximum at that instant, is chosen as the estimate of \( d_i \). This in turn is used to update the probabilities of all the actions.

Comment 3: The number of estimators was arbitrarily chosen as 9 in the above discussion. The actual number chosen depends on the resolution that the designer desires. In the learning schemes, it is not the absolute value of the reward probability that is relevant but the separation of the two highest values of the reward probabilities.

Comment 4: From the foregoing discussion, it is apparent that the MM approach is very similar to the pursuit algorithm. Further, it was also stated that \( \frac{n_i}{N_i} \) used in the latter is, at every stage, the theoretical optimal of the likelihood function. It is therefore necessary to justify the use of multiple models over the simpler pursuit algorithm.

In summary, the following questions need to be answered:
(i) Is the multiple models approach \( \epsilon \)-optimal?
(ii) In what respect is the approach superior to the pursuit algorithm?
(iii) What are the general convergence properties of the scheme?

While work is currently in progress on (iii), we shall attempt to answer questions (i) and (ii) briefly in section III.

III. PROPERTIES OF THE MULTIPLE MODELS APPROACH

As stated in the previous section, our objective is to demonstrate that the MM approach has desirable convergence properties and that though it resembles the pursuit algorithm, it is considerably more robust. The practical demonstration of this is evident from the simulation studies included in sections IV and V. Hence we merely confine our attention in this section to the \( \epsilon \)-optimality of the MM approach and the reasons for its robustness.

A. Proof of \( \epsilon \)-optimality of the MM approach

**Theorem 1** In every stationary random environment a learning automaton using the MM-approach is \( \epsilon \)-optimal. That is, given any \( \epsilon > 0 \), \( \delta > 0 \), there exits \( \eta^* > 0 \) and \( k_0 < \infty \) such that

\[
P[|p_{\text{opt}}(k) - 1| < \epsilon] > 1 - \delta
\]

\( \forall k > k_0, 0 < \eta < \eta^* \)

In (3), \( p_{\text{opt}}(k) \) is the action probability corresponding to the optimal action, and \( \eta \) corresponds to the step size.

**Outline of Proof:** The proof of Theorem 1 proceeds in two stages. First it will be shown that with sufficiently small step size \( \eta \), all actions are chosen sufficient number of times so that \( d_m(k) \) will remain the maximum element of the set \( D(k) = \{d_1(k), \cdots, d_n(k)\} \) after a finite time. Following this, it is shown that this implies convergence of \( p_{\text{opt}}(k) \) as stated in (3).

**Lemma 1** For given constants \( \delta > 0 \) and \( M < \infty \), there exists \( \eta^* > 0 \) and \( N_0 < \infty \) such that using the MM approach for all \( \eta \in (0, \eta^*) \), the probabilities that all actions are chosen at least \( M \) times each before the instant \( k \) is greater than \( (1-\delta) \) for all \( k \geq N_0 \). The proof is long and omitted here.

**Lemma 2** If there exists an index \( \text{"opt"} \) and a time instant \( N_0 < \infty \) such that \( d_{\text{opt}}(k) > d_j(k) \) for all \( j \neq \text{opt} \) and \( k > k_0 \), then \( p_{\text{opt}}(k) \rightarrow 1 \) with probability 1 as \( k \rightarrow \infty \). Consequently,

\[
P[|p_{\text{opt}}(k) - 1| < \epsilon] > 1 - \delta
\]

\( \forall k \geq N_0 \)

**Proof of Theorem 1:** If \( \Delta p(k) \) is defined as \( E[p_{\text{opt}}(k + 1) - p_{\text{opt}}(k)\mid Q(k)] \) where \( Q(k) \) is the state of the system consisting of \( \{p(k), d(k)\} \) then

\[
\Delta p(k) = \eta(1 - p_{\text{opt}}(k)) \quad \forall k \geq N_0
\]

Hence by the Martingale Convergence Theorem (and noting that \( \eta \neq 0 \)), it follows that \( p_{\text{opt}}(k) \rightarrow 1 \) w.p. 1 as \( k \rightarrow \infty \).

**B. Robustness of the MM approach**

In their paper [8], Martin and Tilak argued that the convergence of the running average estimates \( \frac{n_i}{N_i} \) of the reward probabilities in the pursuit algorithm is not monotonic. i.e. the estimate can come arbitrarily close to the true value but diverge from it. This can result in oscillations in the convergence of the action probabilities (see simulation studies in Sections IV and V).

With the MM approach due to quantization (assuming that an appropriate number of models are chosen), whenever there is an error made in the choice of the optimal action, a similar error occurs in the pursuit algorithm. However this is not true in general in reverse (though exceptional cases exist). Hence the robustness of the MM approach will be better than that of pursuit algorithm. Work is currently in progress to make this qualitative statement more precise.

IV. SIMULATION STUDIES

It is generally known that the number of situations in which learning is encountered is extremely large. Hence,
when we are introducing a new and general approach to learning, it is incumbent upon us to test the scheme incrementally from very simple learning situations to very complex cases, if it is to be convincing to a wide audience. This is what is being attempted in this section. The MM approach has been simulated extensively and considerable empirical experience has been gained. In this paper, we first compare in simulation 1, the method with the simple learning automata schemes of the 1960s and 1970s and in simulation 2, with the indirect pursuit algorithm proposed in the 1980s. Simulation 2 is concerned with a feed-forward network. Only one example of each is given for the sake of continuity, since many other examples were presented in [6] when the method was first introduced. Our interest, however, in this section is in situations where feedback is present, and these occur naturally in discrete-state dynamic environments represented by Markov Decision Processes (MDP). In simulations 3, 4, 5, four, five and six node networks were considered. The extension of the methods to continuous dynamic environments will be considered in future papers. In all cases, the problem to be solved is clearly stated and how the different methods being compared perform are presented.

A. Simulation 1: A Learning Automaton

A simple learning automaton has nine actions \( \{\alpha_1, \cdots, \alpha_9\} \) with reward probabilities \( d_1 = 0.12, d_2 = 0.53, d_3 = 0.96, d_4 = 0.2, d_5 = 0.25, d_6 = 0.32, d_7 = 0.44, d_8 = 0.72, d_9 = 0.8 \). The optimal action is \( \alpha_3 \) and has a reward probability \( d_3 = 0.96 \). The next best action is \( \alpha_9 \) with reward probability \( d_9 = 0.8 \). Both the L-RI scheme (with step size 0.05) and the multiple models approach were used to determine the optimal action, and the results are shown in Figure (2).

In the latter case, each action used nine estimates (i.e. total of 81 estimate) to choose the action probabilities at every step.

Comment 1: The MM approach converges with a probability of 0.95 to the optimal action in 250 steps while the learning automaton takes approximately 2500 steps. Hence all future comparisons of the method will be with the pursuit algorithm.

In Figure (3), the choice of the best action in five different trials using the pursuit algorithm are shown, and its non-monotonic behavior is evident in some of them.

B. Simulation 2: Feed-forward Network

A feed-forward network with three layers (0,1,2) is shown in Figure (4). Nine actions \( \{\alpha_1, \alpha_2, \cdots, \alpha_9\} \) can be performed in state \( x_0 \) at layer 0 while two actions can be performed at each of the nodes \( x_1 \) and \( x_2 \) in layer 1.

The action \( \beta_1 \) or \( \beta_2 \) can be performed at \( x_1 \) and actions \( \gamma_1 \) or \( \gamma_2 \) can be performed at \( x_2 \). The actions \( \alpha_i \) can choose either node \( x_1 \) or \( x_2 \), while actions \( \{\beta_1, \beta_2\} \) and \( \{\gamma_1, \gamma_2\} \) can choose nodes \( x_3 \) or \( x_4 \) in layer 2. The transition probability matrices from \( x_0 \) to \( x_1 \) and \( x_2 \) and from \( x_1 \) and \( x_2 \) to \( x_3 \)
and $x_4$ are respectively $P$, $M$, and $N$ where

$$P = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \\ p_{31} & p_{32} \\ p_{41} & p_{42} \\ p_{51} & p_{52} \\ p_{61} & p_{62} \\ p_{71} & p_{72} \\ p_{81} & p_{82} \\ p_{91} & p_{92} \end{bmatrix} = \begin{bmatrix} 0.12 & 0.88 \\ 0.53 & 0.47 \\ 0.96 & 0.04 \\ 0.20 & 0.80 \\ 0.25 & 0.75 \\ 0.32 & 0.68 \\ 0.44 & 0.56 \\ 0.72 & 0.28 \\ 0.80 & 0.20 \end{bmatrix}$$

(6)

$$M = \begin{bmatrix} M_{13} & M_{14} \\ M_{23} & M_{24} \end{bmatrix} = \begin{bmatrix} 0.1 & 0.9 \\ 0.4 & 0.6 \end{bmatrix}$$

(7)

$$N = \begin{bmatrix} N_{13} & N_{14} \\ N_{23} & N_{24} \end{bmatrix} = \begin{bmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \end{bmatrix}$$

(8)

1) Reward: On reaching the states $x_3$ and $x_4$ the system receives rewards of 10 and 20.

2) Objective: To choose actions from the action sets $\alpha = \{\alpha_1, \ldots, \alpha_9\}$, $\beta = \{\beta_1, \beta_2\}$ and $\gamma = \{\gamma_1, \gamma_2\}$ at the three nodes $x_0$, $x_1$ and $x_2$ to optimize the reward.

3) Action Probabilities: At the various modes are updated after states $x_3$ and $x_4$ are reached.

4) Learning Schemes: The multiple models approach was used at each node with 9 models corresponding to each action. The pursuit algorithm was updated at each node with 9 models corresponding to each action, assuming that the transition probabilities are known. It was demonstrated that action $\alpha_3$ at $x_0$, action $\beta_1$ at $x_1$ and action $\gamma_2$ at $x_2$ gave the optimal reward.

5) Optimization: Before carrying out the simulations, the rewards were computed using all possible sequences assuming that the transition probabilities are known. It was demonstrated that action $\alpha_3$ at $x_0$, action $\beta_1$ at $x_1$ and action $\gamma_2$ at $x_2$ gave the optimal reward.

The behavior of action probabilities at layer 0 node $x_0$ using the two methods are shown in Figures (5a) and (5b).

In simulations 3, 4 and 5, feedback was introduced for the first time. All of them represent Markov Decision Processes in which decisions are made at 4, 5 and 6 nodes respectively. Every node represents a state and every node can be accessed by every other node. The transition from one node to another results in a reward. Each state $x_i$ has a finite action set $A_i = \{\alpha_{i1}, \alpha_{i2}, \ldots\}$, and the transition probabilities from any state to any other state under a specified action (unknown), and expressed as a state transition probability function, i.e. if $x_1$ and $x_2$ are two states then $p(s_1, s_2, \alpha)$ is the transition probability from state $x_1$ to state $x_2$ under the action $\alpha \in A_i$.

The objective of the learning agent is to determine a decision rule or policy to maximize the cumulative discounted reward over a finite or infinite time horizon. In all of the simulations, an infinite time horizon was used with a discount factor at every state of $\gamma (0.3, 0.5)$.

Since the networks vary only in the number of nodes, and are completely specified by the number of actions at every node, the transition rewards, the discount factor and the transition probabilities, we shall not describe them in detail here but merely indicate their structure in Figure (6). Due to the space limitations, it is not possible to compare the pursuit algorithm and the multiple models approach in detail. Instead, we present what we consider as the relevant features that need to be emphasized.

C. Simulation 3: a four node network

If $t_{ij}$ stands for the transition probability from node $i$ to $j$, and $T$ is the matrix of transition probabilities, i.e. $T = [t_{ij}]_{i=1,2,3,4}$ and $T_{\alpha_1}$ corresponds to the transition probability for action $\alpha_1$ and $T_{\alpha_2}$ for $\alpha_2$ respectively.

$$T_{\alpha_1} = \begin{bmatrix} 0 & 0.2 & 0.3 & 0.5 \\ 0.4 & 0.3 & 0 & 0.2 \\ 0.4 & 0.5 & 0.3 \\ 0.2 & 0.3 & 0.4 \end{bmatrix}$$

$$T_{\alpha_2} = \begin{bmatrix} 0 & 0.5 & 0.3 & 0.2 \\ 0.2 & 0.5 & 0.3 \\ 0.3 & 0.3 & 0.4 \\ 0.4 & 0.2 & 0.4 \end{bmatrix}$$

From the matrix $T$, it can be observed that all the states are transient states, as by choosing either action 1 or 2, the probability of staying at a specific node is equal to 0. Also, initial rewards are given at each state as $r_1 = 3$, $r_2 = 1$, $r_3 = 1$, $r_4 = 2$. To evaluate the best action to be chosen at each node, the discount reward $v_i$ for each action from its neighbours will be calculated and stored. The sum of current node reward and discount rewards from its neighbours will be used as a criteria for comparing which action should be selected. The mathematical representation of the action would be...
selection procedure at $i^{th}$ node is as following:

$$k_1^i(n) = \sum_{j=1, j \neq i}^4 \hat{d}_{ij}^1(n)(r_j + \gamma v_j(n - 1))$$

$$k_2^i(n) = \sum_{j=1, j \neq i}^4 \hat{d}_{ij}^2(n)(r_j + \gamma v_j(n - 1))$$

(9)

where $k_1^i$ and $k_2^i$ are the discounted rewards at node $i$ for actions 1 and 2. $\hat{d}_{ij}^1$ and $\hat{d}_{ij}^2$ are the estimated transition probabilities from node $i$ to $j$ of actions 1 and 2. $v_j$ is the maximum averaged discounted reward from previous iteration, i.e. $v_j=\max[k_1^j(n-1), k_2^j(n-1)]$. $\gamma$ is the discounted factor for rewards, which is 0.5 in this example. By choosing the action with a larger value of $k_i$, its corresponding action probability $p$ will be updated as described in the previous section.

In the four node network shown in Figure (6a), node 4 has two actions $\alpha_{41}$ and $\alpha_{42}$, of which $\alpha_{42}$ is the optimal. The simulations were repeated 1000 times and the average behavior of the action probabilities is shown in Figure (7). It is seen that the convergence rate of the multiple model approach is faster than the pursuit algorithm, and a considerably smaller fraction of trials converge to the non-optimal value (i.e. probability of convergence to the correct action is higher). These results are found to be typical of those obtained for the other networks.

D. Simulation 4: a five node network

In this case, we are interested in the sample paths of the probabilities of the optimal actions at the various nodes. Due to space limitations, the transition probabilities will be omitted and only simulation results will be given. The plots for the two methods of the optimal action at node 5 are shown in Figure (8).

As shown in the figure, the action probabilities obtained from the pursuit algorithm is unstable, it sometimes reverses back after it reaches 1 or 0. This is mainly the reason that it is not $\epsilon$-optimal. In the MM approach, the action probabilities reaches the correct value monotonically.

E. Simulation 5: a six node network

The two networks considered previously are both fully connected. In this simulation, we consider a case when some of the nodes in the network are not connected to each other, i.e. partially connected network. As all the nodes are not connected, some of the nodes may be totally ignored after many iterations if their rewards are much smaller than others, which will lead to an even longer convergence time compared to the fully connected network case. In this simulation, we show the evolution of the averaged rewards for actions 1 and 2 at node 6 in Figure (9). (Initial rewards given at each node are $(r_1 = 0.5, r_2 = 0.1, r_3 = 0.3, r_4 = 0.2, r_5 = 0.8, r_6 = 0.6)$.

From the simulation, it is easily observed that multiple models give a faster convergence of action rewards compared to conventional pursuit algorithm, though the node is sparsely connected to the rest of network.
V. APPLICATION TO BLACKJACK

In section IV, the new approach was compared with a direct method (a Linear Reward-Inaction scheme of learning automata), the indirect pursuit algorithm in feed-forward networks, and with Markov Decision Processes (MDP). Currently, the method is being applied to many real problems posed by workers in the field of reinforcement learning. One such problem is Blackjack. Since all readers may not be familiar with it, we first give a brief description as obtained from Google, state the mathematical problem and provide the results for comparison purposes.

Blackjack (or twenty-one) is the most widely played casino banking game in the world. The game is between a player and a dealer. It is played with one or more decks of 52 cards, but in the present case we shall assume that multiple decks are used to prevent counting of cards. The objective is to beat the dealer as follows:

(i) get 21 points on the first two cards (blackjack) without a dealer blackjack
(ii) reach a final score higher than the dealer without exceeding 21, or
(iii) let the dealer draw additional cards until his or her hand exceeds 21.

The player is dealt a two card hand and adds together the value of the cards. Face cards are counted as ten points and an ace can be either one or ten. All other cards assume their numeric values.

A. The Game

In the initial state, both the dealer and the player are dealt two cards. Both the players cards are visible while only one card of the dealer is visible. At every state, the player decides to "hit" or "stay". If the player hits, an additional card is drawn and the value of the card is added to the total. In a given round, the player or dealer wins by having a score of 21, or by having the higher score that is less than 21. Scoring higher than 21 (going bust) results in a loss. A player having 21 or fewer points when the dealer busts, wins the round. The dealer must hit until the cards total 17 or more points.

B. The State Space

The state space consists of the player's sum, the dealer's card which can be seen and whether the player has an ace. In all, there are two hundred states. The objective is for a learning algorithm to decide what strategy to use (hit or stay) in each state.

C. The Learning Schemes

The following three learning schemes were compared. The game is played repeatedly using each of the competing algorithms, and the strategies are updated:

(i) the $\epsilon$-greedy algorithm in Sutton and Barto's book "Reinforcement Learning- An introduction"
(ii) the pursuit algorithm and
(iii) the approach based on multiple models.

It is known that the optimal winning percentage is 42.4%.

The questions we are interested in answering are how close the learning algorithms come to the optimal, how rapidly they approach the value, and the nature of the convergence of the strategies.

D. Summary of the results

a) The $\epsilon$-greedy algorithm converges to 14% less than the optimal value.
b) In some cases the pursuit algorithm is highly oscillatory (as observed in the previous section) in the initial stages. The multiple model approach exhibits such behavior much more infrequently.
c) In some cases, the pursuit algorithm converges to a suboptimal value in a jagged fashion. The multiple model based approach is smooth, and convergence is generally monotonic.

E. Simulation

The results are shown in Figure (10). The $\epsilon$-greedy algorithm is the least accurate, the multiple model method is smooth and has the largest asymptotic value, while pursuit algorithm exhibits oscillatory behavior before converging to a value close to the latter.
VI. CONCLUSION

The multiple model approach introduced in this paper appears to have enormous potential in stochastic learning problems. It has desirable convergence properties. It is at least as fast as the pursuit algorithm, but substantially more robust. The authors firmly believe that it will be far superior to the latter in more complex and interconnected systems.

Only fixed models (estimates) were used in the simulation studies. Adaptive models (estimates) have also been investigated and appear to complement the fixed models. These will be reported in future papers.

Since the acid test for a learning scheme is how it performs in real applications, work is currently in progress to test the approaches on difficult problems that are being addressed by members of the reinforcement learning community.

REFERENCES