It is simple to check that the matrix $(A_1^{(1)}A_2^{(2)}A_2^{(3)})$ is not Hurwitz, and, hence, it follows from Theorem 4.1 that the systems $\Sigma_{A_1}, \Sigma_{A_2}$ do not have a common linear copositive Lyapunov function.

The above example shows that two stable positive LTI systems whose system matrices differ by a rank one matrix need not in general have a common linear copositive Lyapunov function. However, the next corollary provides a simple sufficient condition for the existence of a common linear copositive Lyapunov function for this case.

Corollary 5.2: Let $A_1, A_2 = A_1 + B$ be Metzler, Hurwitz matrices in $\mathbb{R}^{n\times n}$, with $\operatorname{rank}(B) = 1$. For each $i \in \{1, \ldots, n\}$, let $T_i \in \mathbb{R}^{n\times n}$ be the matrix given by

$$T_i^{(j)} = \begin{cases} B^{(j)}, & \text{if } j = i \\ A_1^{(j)}, & \text{if } j \neq i. \end{cases}$$

Then the positive LTI systems, $\Sigma_{A_1}, \Sigma_{A_2}$ have a common linear copositive Lyapunov function if for $1 \leq i \leq n$, either $\operatorname{sign}(\det(T_i)) = (-1)^n$ or $\operatorname{sign}(\det(T_i)) = 0$.

Proof: Suppose that for $1 \leq i \leq n$, either $\operatorname{sign}(\det(T_i)) = (-1)^n$ or $\operatorname{sign}(\det(T_i)) = 0$. As $\operatorname{rank}(B) = 1$, we can write $B = bc^T$ for column vectors $b, c \in \mathbb{R}^n$. (Thus, all columns of B are scalar multiples of each other). It follows from this, and the linear dependence of the determinant function on each column, that for any $A \in \mathcal{S}(A_1, A_2)$, there is some set of indices $\{i_1, \ldots, i_k\} \subset \{1, \ldots, n\}$ such that

$$\det(A) = \det(A_1) + \sum_{j=1}^{k} \det(T_{i_j}). \tag{13}$$

Hence, as $\operatorname{sign}(\det(T_i))$ is either $(-1)^n$ or 0 for $1 \leq i \leq n$, it follows that $\operatorname{sign}(\det(A)) = (-1)^n$ for all $A \in \mathcal{S}(A_1, A_2)$. Corollary 4.2 immediately implies that $\Sigma_{A_1}, \Sigma_{A_2}$ have a common linear copositive Lyapunov function as claimed.

VI. Conclusion

In this paper, we have presented a method for determining whether or not a given switched positive continuous time linear system is exponentially stable. Our approach is based upon determining verifiable conditions for the existence of a common copositive linear Lyapunov function for a pair of positive LTI systems. Future work will involve extending this result to arbitrary finite sets of such LTI systems, and developing synthesis procedures to exploit our result for the design of stable switched positive systems.

References

[1] A. Berman and R. J. Plemmon, "Non-negative matrices in the mathematical sciences," *SIAM Classics Appl. Math.*, 1994.

[2] L. Farina and S. Rinaldi, *Positive Linear Systems*. New York: Wiley, 2000.

[3] L. Gurvits, R. Shorten, and O. Mason, "On the stability of switched positive linear systems," *IEEE Trans. Autom. Control*, to be published.

[4] R. Horn and C. Johnson, *Topics in Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1991.

[5] A. Jadbabaie, J. Lin, and A. S. Morse, "Co-ordination of groups of mobile autonomous agents using nearest neighbour rules," *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 988–1001, Jun. 2003.

[6] D. Leith and P. Clifford, "Convergence of distributed learning algorithms for optimal wireless channel allocation," presented at the IEEE Conf. Decision and Control, 2006.

[7] O. Mason and R. Shorten, "On the simultaneous diagonal stability of a pair of positive linear systems," *Linear Algebra Appl.*, vol. 413, pp. 13–23, 2006.

[8] A. Paul, M. Akar, U. Mitra, and M. Safonov, "A switched system model for stability analysis of distributed power control algorithms for cellular communications," presented at the American Control Conf., 2004.

[9] R. T. Rockafellar, *Convex Analysis*. Princeton, NJ: Princeton Univ. Press, 1970.

[10] R. N. Shorten, F. Wirth, and D. Leith, "A positive systems model of TCP-like congestion control: Asymptotic results," *IEEE Trans. Netw.*, vol. 14, no. 3, pp. 616–629, Mar. 2006.

Recursive Learning Automata Approach to Markov Decision Processes

Hyeong Soo Chang, Michael C. Fu, Jiaqiao Hu, and Steven I. Marcus

***Abstract*—In this note, we present a sampling algorithm, called recursive automata sampling algorithm (RASA), for control of finite-horizon Markov decision processes (MDPs). By extending in a recursive manner Sastry's learning automata pursuit algorithm designed for solving nonsequential stochastic optimization problems, RASA returns an estimate of both the optimal action from a given state and the corresponding optimal value. Based on the finite-time analysis of the pursuit algorithm by Rajaraman and Sastry, we provide an analysis for the finite-time behavior of RASA. Specifically, for a given initial state, we derive the following probability bounds as a function of the number of samples: 1) a lower bound on the probability that RASA will sample the optimal action and 2) an upper bound on the probability that the deviation between the true optimal value and the RASA estimate exceeds a given error.**

***Index Terms*—Learning automata, Markov decision process (MDP), sampling.**

I. Introduction

Consider the following finite-horizon ($H < \infty$) Markov decision processes (MDP) model (see, e.g., [7] and [10]): $x_{i+1} = f(x_i, a_i, w_i)$ for $i = 0, 1, 2, \ldots, H-1$, where x_i is a random variable ranging over a (possibly infinite) state set X giving the state at stage i, a_i is the control to be chosen from a finite action set A at stage i, w_i is a random disturbance uniformly and independently selected from $[0, 1]$ at stage i, representing the uncertainty in the system, and $f : X \times A \times [0, 1] \to X$ is the next-state function.

Manuscript received July 5, 2006; revised December 18, 2006. Recommended by Associate Editor I. Paschalidis. This work was supported in part by the National Science Foundation under Grant DMI-0323220, the Air Force Office of Scientific Research under Grant FA95500410210, and the Department of Defense. The work of H. S. Chang was supported by the Korea Research Foundation under Grant KRF-2005-041-C00066.

H. S. Chang is with the Department of Computer Science and Engineering, Sogang University, Seoul 121-742, Korea (e-mail: hschang@sogang.ac.kr).

M. C. Fu is with the Robert H. Smith School of Business and the Institute for Systems Research, the University of Maryland, College Park, MD 20742 USA (e-mail: mfu@rhsmith.umd.edu).

J. Hu is with the Department of Applied Mathematics and Statistics, State University of New York (SUNY), Stony Brook, NY 11794-3600 USA (e-mail: jqhu@ams.sunysb.edu).

S. I. Marcus is with the Department of Electrical and Computer Engineering and the Institute for Systems Research, the University of Maryland, College Park, MD 20742 USA (e-mail: marcus@eng.umd.edu).

Digital Object Identifier 10.1109/TAC.2007.900859

Let Π be the set of all possible nonstationary Markovian policies $\pi = \{\pi_i \mid \pi_i : X \to A, i = 0, \ldots, H-1\}$. Defining the reward-to-go value function of $\pi \in \Pi$ for state x in stage i by

$$V_i^{\pi}(x) = E_w \left[\sum_{t=i}^{H-1} R(x_t, \pi_t(x_t), w_t) \mid x_i = x \right]$$

where $x \in X, w = (w_i, w_{i+1}, \ldots, w_{H-1}), w_j \sim U(0,1), j = i, \ldots, H-1$, with a bounded nonnegative reward function $R : X \times A \times [0,1] \to \mathcal{R}^+$, $V_H^{\pi}(x) = 0$ for all $x \in X$, and $x_t = f(x_{t-1}, \pi_{t-1}(x_{t-1}), w_{t-1})$ a random variable denoting the state at stage t following policy π, the goal of this note is to estimate the optimal reward-to-go value $V_0^*(x_0) = \sup_{\pi \in \Pi} V_0^{\pi}(x_0)$ and obtain an approximately optimal action for a *given fixed initial state* $x_0 \in X$. R is bounded such that $R_{\max} := \sup_{x \in X, a \in A, w \in [0,1]} R(x, a, w) < \infty$ and, for simplicity, it is assumed that every action in A is admissible at every state and the same random number is associated with the reward and transition functions.

This note presents a sampling algorithm, called recursive automata sampling algorithm (RASA), which extends in a recursive manner (for sequential decisions) the pursuit algorithm by Sastry [13] in the context of solving the problem for MDPs. The running time complexity of RASA is $O((\max_i K_i)^H)$, where K_i is the total number of samples that are used per state sampled at stage i, and the space complexity is $O(|A|(\max_i K_i)^H)$. The complexities are independent of the state–space size $|X|$ but exponentially dependent on the horizon size H.

The pursuit algorithm is based on well-known learning automata (LA) [17], [14] for solving (nonsequential) stochastic optimization problems. A learning automaton is associated with a finite set of actions (candidate solutions) and updates a probability distribution over the set by iterative interaction with an environment and takes (samples) an action according to the newly updated distribution. The environment provides a certain reaction (reward) to the action taken by the automaton, where the reaction is random and the distribution is unknown to the automaton. The automaton's aim is to learn to choose the optimal action that yields the highest average reward. In the pursuit algorithm, the automaton *pursues* the current best optimal action by increasing the probability of sampling that action while decreasing the probabilities of all other actions.

As LA are well-known *adaptive* decision-making devices operating in unknown random environments, RASA's sampling process of taking an action at the sampled state is adaptive at each stage. At each sampled state at a stage, a fixed sampling budget is allocated among feasible actions, and the budget is used with the current probability estimate for the optimal action. RASA builds a sampled tree in a recursive manner to estimate the optimal value $V_0^*(x_0)$ at an initial state x_0 in a bottom-up fashion and makes use of an adaptive sampling scheme of the action space while building the tree. Based on the finite-time analysis of the pursuit algorithm by Rajaraman and Sastry [11], we analyze the finite-time behavior of RASA, providing the following in terms of the sampling parameters of RASA for a given x_0: 1) a lower bound on the probability that the optimal action is sampled, and 2) an upper bound on the probability that the difference between the estimate of $V_0^*(x_0)$ and the true value of $V_0^*(x_0)$ exceeds a given error.

Recent theoretical progress (e.g., [4]–[6]) on the development of (simulation-based) algorithms has led to practically viable approaches to solving large finite-horizon MDPs. For example, the "simulated annealing multiplicative weights" algorithm in [6] also works by updating a probability distribution, but over a *restricted policy space* rather than the entire action space. More closely related to RASA is the adaptive multistage sampling (AMS) algorithm [4], which also uses adaptive sampling in a manner similar to RASA and has similar time and space complexities, but the sampling in AMS is based on the exploration–exploitation process of multiarmed bandits from regret analysis, and the performance analysis of AMS is given in terms of the expected bias for *finite* state–space problems, establishing an asymptotically unbiased estimate of the optimal value. In contrast, RASA's sampling is based on the probability estimate for the optimal action, and probability bounds are provided for the performance analysis of RASA. The convergence property of RASA is stronger than AMS in that *convergence in probability* is established and *infinite* state–space problems are considered. Furthermore, the analysis of RASA includes a correction of Lemma 3.2 in [11] (see Theorem 3.1 in Section III).

The book by Poznyak *et al.* [9] provides a good survey on using LA for solving controlled (ergodic) Markov chains in a model-free reinforcement learning (RL) [16] framework for a loss function defined on the chains. Each automaton is associated with a state and updates certain functions (including the probability distribution over the action space) at each iteration of various learning algorithms. Wheeler and Narendra [18] consider controlling ergodic Markov chains for *infinite-horizon* average reward within a similar RL framework. As far as the authors are aware, for all of the LA approaches proposed in the literature, the number of automata grows with the size of the state–space, and thus, the corresponding updating procedure becomes cumbersome, with the exception of Wheeler and Narendra's algorithm, where only the currently visited automaton (state) updates relevant functions. However, since Wheeler and Narendra's algorithm is designed for infinite-horizon problems, it is not clear how it should be modified for the finite-horizon setting, as there are a number of possible alternatives. In terms of theoretical results, the results available for LA-based approaches only prove the *convergence* of the algorithms for *finite* state–space problems, whereas we also provide finite-time *bounds* in the *infinite* state–space setting.

Santharam and Sastry [12] provide an adaptive method for solving infinite-horizon MDPs via a stochastic neural network model in the RL framework. For each state, there are $|A|$-units of nodes associated with the estimates of "state-action values." As in [18], the probability distribution over the action space at an observed state is updated with an observed payoff. Even though they provide a probabilistic performance guarantee for their approach, the network size grows with the sizes of the state and action spaces. Ahamed *et al.* [1] propose an RL-based approach for solving infinite-horizon discounted reward MDPs by incorporating the pursuit algorithm [13] into the exploration–exploitation process of the learning. At each learning time step, the greedy action with respect to the current optimal "state-action value" estimate is pursued for updating the probability distribution over the action space of the currently visited state.

In neurodynamic programming approach [3], the state-action value can be estimated by a class of parameterized functions with a suitable learning algorithm for obtaining the parameters to tackle infinite state spaces. The success of the function–approximation approach depends heavily on the choice of a good architecture, which is generally problem dependent. Furthermore, the quality of the approximation is often difficult to gauge in terms of useful theoretical error bounds [5].

Jain and Varaiya [8] provide a *uniform* bound on the empirical performance of policies within a simulation model of (partially observable) MDPs, which would be useful in many contexts. However, the main interest here is an algorithm and its sampling complexity for estimating the *optimal* value/policy, and not the value function over the *entire* set of feasible policies.

This note is organized as follows. We present RASA in Section II and analyze it in Section III, with the proofs presented in the appendices. A numerical example is provided in Section IV. We conclude our note in Section V.

Recursive Automata Sampling Algorithm (RASA)

- **Input**: stage $i \neq H$, state $x \in X$, $K_i > 0$, $\mu_i \in (0,1)$. (If $i = H$, then **Exit** immediately, returning $V_H^{K_H}(x) = 0$.)
- **Initialization:** Set $P_x(0)(a) = 1/|A|, N_a^i(x) = 0, M_i^{K_i}(x,a) = 0 \ \ \forall a \in A; k = 0$.
- **Loop** until $k = K_i$:
  - Sample $a(k) \sim P_x(k)$, $w_k \sim U(0,1)$.
  - Update Q-value estimate for $a = a(k)$ only:

$$M_i^{K_i}(x,a(k)) \leftarrow M_i^{K_i}(x,a(k)) + R(x,a(k),w_k) + \hat{V}_{i+1}^{K_{i+1}}(f(x,a(k),w_k)) \quad (1)$$

$$N_{a(k)}^i(x) \leftarrow N_{a(k)}^i(x) + 1$$

$$\hat{Q}_i^{K_i}(x,a(k)) \leftarrow \frac{M_i^{K_i}(x,a(k))}{N_{a(k)}^i(x)}.$$

  - Update optimal action estimate: (ties broken arbitrarily)

$$\hat{a}^* = \arg\max_{a \in A} \hat{Q}_i^{K_i}(x,a). \quad (2)$$

  - Update probability distribution over action space:

$$P_x(k+1)(a) \leftarrow (1-\mu_i)P_x(k)(a) + \mu_i I\{a = \hat{a}^*\} \ \ \forall a \in A, \quad (3)$$

    where $I\{\cdot\}$ denotes the indicator function.
  - $k \leftarrow k+1$.
- **Exit:** Return $\hat{V}_i^{K_i}(x) = \hat{Q}_i^{K_i}(x,\hat{a}^*)$.

Fig. 1. RASA description.

II. Algorithm

It is well known (see, e.g., [10]) that V_i^* can be written recursively as follows: for all $x \in X$ and $i = 0, \ldots, H-1$

$$V_i^*(x) = \max_{a \in A} Q_i^*(x,a)$$
$$Q_i^*(x,a) = E_w[R(x,a,w) + V_{i+1}^*(f(x,a,w))]$$

$w \sim U(0,1)$ and $V_H^*(x) = 0, x \in X$. We provide a high-level description of RASA in Fig. 1 to estimate $V_0^*(x_0)$ for a given initial state x_0 via this set of equations. The inputs to RASA are the stage i, a state $x \in X$, and sampling parameters $K_i > 0$, $\mu_i \in (0,1)$, and the output of RASA is $\hat{V}_i^{K_i}(x)$, the estimate of $V_i^*(x)$, where $\hat{V}_H^{K_H}(x) = V_H^{K_H}(x) = 0 \forall K_H, x \in X$. Whenever $\hat{V}_{i'}^{K_{i'}}(y)$ (for stage $i' > i$ and state y) is encountered in the **Loop** portion of RASA, a recursive call to RASA is required [at (1)]. The initial call to RASA is done with stage $i = 0$, the initial state x_0, K_0, and μ_0, and every sampling is independent of previous samplings.

Basically, RASA builds a sampled tree of depth H with the root node being the initial state x_0 at stage 0 and a branching factor of K_i at each level i (level 0 corresponds to the root). The root node x_0 corresponds to an automaton and initializes the probability distribution over the action space P_{x_0} as the uniform distribution (refer to **Initialization** in RASA). The x_0-automaton then samples an action and a random number (an action and a random number together corresponding to an edge in the tree) independently with respect to (w.r.t.) the current probability distribution $P_{x_0}(k)$ and $U(0,1)$, respectively, at each iteration $k = 0, \ldots, K_0 - 1$. If action $a(k) \in A$ is sampled, the count variable $N_{a(k)}^0(x_0)$ is incremented. From the sampled action $a(k)$ and the random number w_k, the automaton obtains an environment response $R(x_0,a(k),w_k) + \hat{V}_1^{K_1}(f(x_0,a(k),w_k))$. Then, by averaging the responses obtained for each action $a \in A$ such that $N_a^0(x_0) > 0$, the x_0-automaton computes a sample mean for $Q_0^*(x_0,a)$: $\hat{Q}_0^{K_0}(x_0,a) = N_a^0(x_0)^{-1} \sum_{j:a(j)=a} R(x_0,a,w_j) + \hat{V}_1^{K_1}(f(x_0,a,w_j))$, where $\sum_{a \in A} N_a^0(x_0) = K_0$. At each iteration k, the x_0-automaton obtains an estimate of the optimal action by taking the action that achieves the current best Q-value [cf. (2)] and updates the probability distribution $P_{x_0}(k)$ in the direction of the current estimate of the optimal action $\hat{a}^*$ [cf. (3)]. In other words, the automaton *pursues* the current best action, and hence, the nonrecursive one-stage version of this algorithm is called the pursuit algorithm [11]. After K_0 iterations, RASA estimates the optimal value $V_0^*(x_0)$ by $\hat{V}_0^{K_0}(x_0) = \hat{Q}_0^{K_0}(x_0,\hat{a}^*)$.

The overall estimate procedure is replicated recursively at those sampled states (corresponding to nodes in level 1 of the tree) $f(x_0,a(k),w_k)$-automata, where the estimates returned from those states $\hat{V}_1^{K_1}(f(x_0,a(k),w_k))$ comprise the environment responses for the x_0-automaton.

The running time complexity of RASA is $O(K^H)$ with $K = \max_i K_i$, and the space complexity is $O(|A|(\max_i K_i)^H)$, independent of the state–space size. (For some performance guarantees, the value K depends on the size of the action space. See Section III.)

III. Analysis of RASA

All of the estimated $\hat{V}^{K_i}$- and $\hat{Q}^{K_i}$-values in this section refer to the values at the **Exit** in the algorithm. The proofs of all lemmas and theorems are given in the appendices. Lemma 3.1 stated within our contexts is a direct application of [11, Lemma 3.1]. It provides a probability bound on the estimate of the Q^*-value relative to the true Q^*-value when the estimate of the Q^*-value is obtained under the assumption that the optimal value for the remaining horizon is known (so that the recursive call is not required).

Lemma 3.1: Given $\delta \in (0,1)$ and positive integer N such that $6 \leq N < \infty$, consider running the one-stage nonrecursive RASA obtained by replacing (1) by

$$M_i^{K_i}(x,a) \leftarrow M_i^{K_i}(x,a) + R(x,a,w_k) + V_{i+1}^*(f(x,a,w_k)) \quad (4)$$

with $K_i > \bar{\lambda}(N,\delta)$ and $0 < \mu_i < \bar{\mu}_i(N,\delta)$, where with $l = 2|A|/(2|A|-1)$

$$\bar{\lambda}(N,\delta) = \left\lceil \frac{2N}{\ln l} \ln \left\lceil \frac{Nl}{\ln l} \left(\frac{N}{\delta} \right)^{\frac{1}{N}} \right\rceil \right\rceil$$
$$\bar{\mu}_i(N,\delta) = 1 - 2^{-1/\bar{\lambda}(N,\delta)}.$$

Then, for each action $a \in A$, we have

$$P\left(\sum_{k=0}^{K_i} I\{a(k) = a\} \leq N \right) < \delta.$$

The following theorem is a corrected version of [11, Lemma 3.2]. The Hoeffding's inequality was incorrectly applied (see [11, eq. (3.19) and (3.21), p. 594]). Although $\{\beta(m_j^i), j = 1,2,\cdots\}$ is an independent identically distributed (i.i.d.) sequence, conditional on $Y_i(k)$ (which is a stopping time for the sequence), $\{\beta(m_j^i), j = 1,2,\cdots\}$ is no longer an i.i.d. sequence.

Theorem 3.1: Let $\{X_i, i = 1,2,\ldots\}$ be a sequence of i.i.d. nonnegative uniformly bounded random variables, with $0 \leq X_i \leq D$ and $E[X_i] = \mu \ \forall i$, and let $M \in \mathcal{Z}^+$ be a positive integer-valued random variable bounded by L. Then, for any given $\epsilon > 0$ and $n \in \mathcal{Z}^+$, we have

$$P\left(\left| \frac{1}{M} \sum_{i=1}^{M} X_i - \mu \right| \geq \epsilon, M \geq n \right) \leq 2e^{-n\left(\frac{D+\epsilon}{D} \ln \frac{D+\epsilon}{D} - \frac{\epsilon}{D}\right)}.$$

Lemma 3. 2 provides a probability bound on the $\hat{Q}_i^{K_i}$-estimate relative to the Q^*-value as a function of the number of samples for a given stage i, for the one-stage nonrecursive RASA.

Lemma 3.2: Consider a fixed $i, x \in X, \epsilon > 0$, and $\delta \in (0,1)$, and the nonrecursive RASA obtained by replacing (1) by (4). Assume $K_i > \lambda(\epsilon, \delta)$ and $0 < \mu_i < \mu_i^*(\epsilon, \delta)$, where

$$\lambda(\epsilon,\delta) = \left\lceil \frac{2M_{\epsilon,\delta}}{\ln l} \ln \left[\frac{lM_{\epsilon,\delta}}{\ln l} \left(\frac{2M_{\epsilon,\delta}}{\delta} \right)^{\frac{1}{M_{\epsilon,\delta}}} \right] \right\rceil$$

with $l = 2|A|/(2|A| - 1)$

$$M_{\epsilon,\delta} = \max\left\{6, \left\lceil \frac{R_{\max} H \ln(4/\delta)}{(R_{\max} H + \epsilon)\ln((R_{\max} H + \epsilon)/R_{\max} H) - \epsilon} \right\rceil \right\}$$

and $\mu_i^*(\epsilon,\delta) = 1 - 2^{-1/K_i}$. Then, for all $a \in A$ at the **Exit** step

$$P\left(\left| \hat{Q}_i^{K_i}(x,a) - Q_i^*(x,a) \right| \geq \epsilon \right) < \delta.$$

We now make an assumption *for the purpose of the analysis*. The assumption states that at each stage, the optimal action is unique. In other words, the given MDP has a unique optimal policy. We will give a remark on this in Section V.

Assumption 3.1: $\theta := \inf_{x \in X, i=0,\ldots,H-1} \theta_i(x) > 0$, where $\theta_i(x) = Q_i^*(x,a^*) - \max_{a \neq a^*} Q_i^*(x,a), x \in X, i = 0,1,\ldots,H-1$, where $a^* = \arg\max_{a \in A} Q_i^*(x,a^*)$.

Given $\delta_i \in (0,1), i = 0,\ldots,H-1$, define

$$\rho := (1-\delta_0) \prod_{i=1}^{H-1} (1-\delta_i)^{\prod_{j=1}^{i} K_j} < 1. \tag{5}$$

Lemma 3.3: Assume that Assumption 3.1 holds. Select $K_i > \lambda(\theta/2^{i+2}, \delta_i)$ and $0 < \mu_i < \mu_i^* = 1 - 2^{-1/K_i}$ for a given $\delta_i \in (0,1), i = 0,\ldots,H-1$. Then, under RASA with ρ in (5), at the **Exit** step

$$P\left(\left| \hat{V}_0^{K_0}(x_0) - V_0^*(x_0) \right| > \frac{\theta}{2} \right) < 1 - \rho.$$

We now present the main result on the probability bounds as a function of the number of samples for a given initial state: 1) a lower bound on the probability that RASA will sample the optimal action and 2) an upper bound on the probability that the deviation between the true optimal value and the RASA estimate exceeds a given error.

Theorem 3.2: Assume that Assumption 3.1 holds.

1) Given $\delta_i \in (0,1), i = 0,\ldots,H-1$, select $K_i > \lambda(\theta/2^{i+2}, \delta_i)$ and $0 < \mu_i < \mu_i^* = 1 - 2^{-1/K_i}, i = 1,\ldots,H-1$. If $K_0 > \lambda(\theta/4, \delta_0) + \lceil \ln \epsilon / \ln(1-\mu_0^*) \rceil$, and $0 < \mu_0 < \mu_0^* = 1 - 2^{-1/\lambda(\theta/4,\delta_0)}$, then under RASA with ρ in (5), for all $\epsilon \in (0,1)$, $P(P_{x_0}(K_0)(a^*) > 1-\epsilon) > \rho$, where $a^* = \arg\max_{a \in A} Q_0^*(x_0,a)$.
2) Given $\delta_i \in (0,1), i = 0,\ldots,H-1$ and $\epsilon \in (0,\theta]$, select $K_i > \lambda(\epsilon/2^{i+2}, \delta_i), 0 < \mu_i < \mu_i^* = 1 - 2^{-1/K_i}, i = 0,\ldots,H-1$. Then, under RASA with ρ in (5)

$$P\left(\left| \hat{V}_0^{K_0}(x_0) - V_0^*(x_0) \right| > \frac{\epsilon}{2} \right) < 1 - \rho.$$

IV. Numerical Example

To illustrate the performance of RASA, we consider a finite-horizon inventory control problem with lost sales and zero-order lead time. At each time period t, given the current inventory level $x_t \in \{0,1,\ldots,M\}$, where M is the maximum inventory capacity, an order in the amount of a_t is placed and received, and a demand D_t is realized. The system dynamics evolve according to the following equation: $x_{t+1} = \max\{0, x_t + a_t - D_t\}$. The goal is to minimize the expected total cost from a given initial state over the set of all nonstationary Markovian policies, i.e., $\min_{\pi \in \Pi} E[\sum_{t=0}^{H-1} (cI\{\pi_t(x_t) > 0\} + h\max\{0, x_{t+1}\} + p\max\{0, -x_{t+1}\}) | x_0 = x]$, where c is the fixed setup cost per order, h is the per period per unit inventory holding cost, and p is the per period per unit demand lost penalty cost.

TABLE I
Performance of RASA, AMS, and NMS on the Inventory Control Problem (Mean of 25 Simulation Replications, With Standard Errors in Parentheses)

(c,p)	Optimal	K	RASA (std err)	AMS (std err)	NMS (std err)
c=0	7.50	10	6.57(0.21)	6.17(0.20)	4.39(0.24)
p=1		20	6.92(0.11)	7.05(0.11)	5.84(0.16)
		40	7.23(0.08)	7.42(0.06)	6.66(0.13)
		60	7.37(0.07)	7.48(0.05)	6.84(0.08)
c=5	25.998	10	23.33(0.27)	23.21(0.44)	18.58(0.49)
p=10		20	24.84(0.25)	25.84(0.20)	22.24(0.38)
		40	25.51(0.12)	26.24(0.11)	23.93(0.26)
		60	25.86(0.09)	26.22(0.09)	24.72(0.18)

In our simulation experiments, we considered the following parameter setting: $H = 3, M = 20, h = 1$, setup cost $c = 0$ and $c = 5$, penalty cost $p = 1$ and $p = 10$, order amount $a_t \in \{0,2,4,6,8,10\}$ for all $t = 1,\ldots,H-1$, and the demand D_t is a discrete random variable with the discrete uniform distribution $\mathrm{DU}(0,9)$.

For comparison purposes, we also applied the AMS algorithm and a nonadaptive multistage sampling (NMS) algorithm to the previous problem. Similar to RASA, both AMS and NMS are simulation-tree-based methods. However, the difference between these algorithms is in the way the actions are sampled at each decision period: both RASA and AMS sample actions in an adaptive manner, whereas NMS simply samples each action for a fixed number of times (cf. [4] for a detailed description). For simplicity, the number of samples at each stage K_i are taken to be the same for all $i = 1,\ldots,H-1$, denoted by a single variable K. Thus, the input parameter μ_i in RASA is chosen to be $\mu_i = 1 - 2^{-1/K}$, independent of stage i. In NMS, whenever a state x is visited, each admissible action at x is sampled for $\lceil K/|A(x)| \rceil$ times, where $A(x)$ indicates the set of admissible actions at state x.

For each test case, we performed 25 independent simulation runs using each of the three algorithms. The results are reported in Table I and in Fig. 2, which plots average performances of the algorithms as a function of the total number of periods simulated. The results indicate convergence of all three algorithms, with RASA and AMS providing superior performance over NMS. In both cases, AMS shows a faster empirical convergence rate, although RASA yields comparable performance when the sample size K is large. For the $c = 5, p = 10$ case, RASA slightly outperforms AMS (which is biased high) after about 1.2×10^5 simulation periods.

V. Concluding Remarks

From the statements of Lemma 3.3 and Theorem 3.2, the performance of RASA depends on the value of θ. If $\theta_i(x)$ is very small or even 0 (failing to satisfy Assumption 3.1) for some $x \in X$, RASA will have a hard time distinguishing between the optimal action and the second best action or multiple optimal actions *if x is in the sampled tree of* RASA. In general, the larger θ is, the more effective the algorithm will be (the smaller the sampling complexity). Therefore, in the actual implementation of RASA, if multiple actions' performances

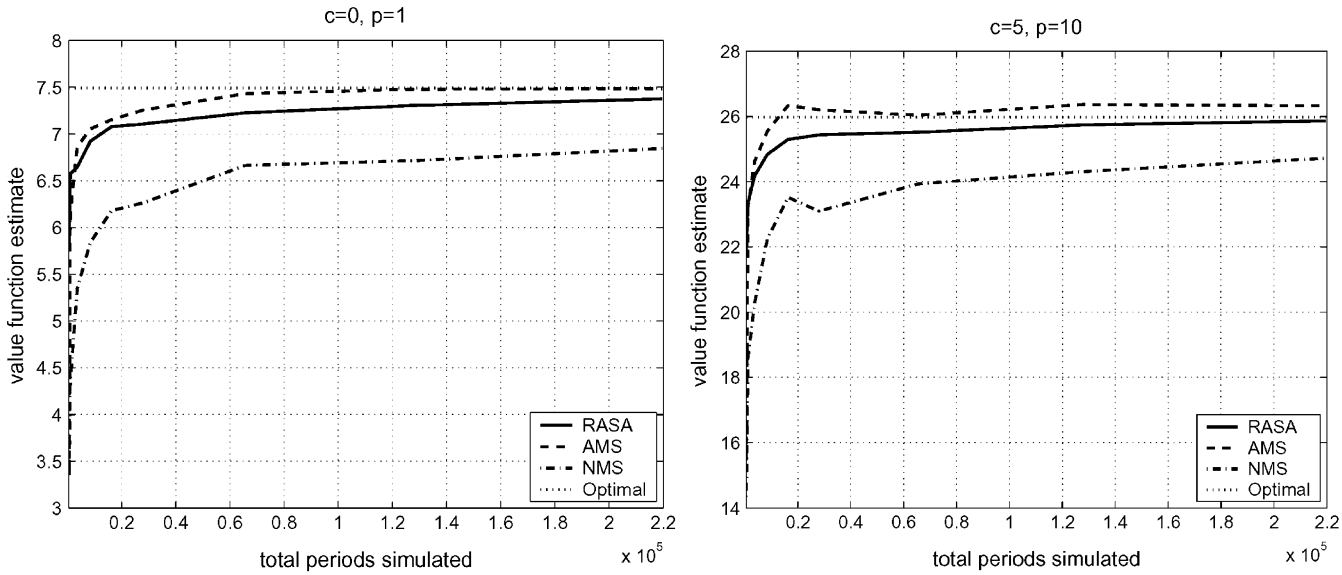

Fig. 2. Average performance (mean of 25 simulation replications of RASA, AMS, and NMS on the inventory control problem with $(c = 0, p = 1)$ and $(c = 5, p = 10)$.

are very close after "enough" iterations in **Loop**, it would be advisable to keep only one action among the competitive actions (transferring the probability mass). The parameter θ can thus be viewed as a measure of problem difficulty. Furthermore, to achieve a certain approximation guarantee at the root level of the sampled tree (i.e., the quality of $\hat{V}_0^{K_0}(x_0)$), we need a *geometric* increase in the accuracies of the optimal reward-to-go values for the sampled states at the lower levels, making it necessary that the total number of samples at the lower levels increase geometrically (K_i depends on $2^{i+2}/\theta$). This is because the estimate error of $V_i^*(x_i)$ for some $x_i \in X$ affects the estimate of the sampled states in the higher levels in a recursive manner (the error in a level "adds up recursively"). However, the probability bounds in Theorem 3.2 are obtained with coarse estimation of various parameters/terms. For example, we used the worst case values of $\theta_i(x), x \in X, i = 0, \ldots, H-1$ and $(R_{\max} H)^2$ for bounding $\sup_{x \in X} V_i^*(x), i = 0, \ldots, H-1$, and used conservative bounds in (11) and in relating the probability bounds for the estimates at the two adjacent levels. Considering this, the performance of RASA should probably be more effective in practice than the analysis indicates here.

Because RASA is designed for possibly infinite state–space problems, it can be used for solving finite-horizon partially observable Markov decision process (POMDP) with finite state, action, and observation spaces, as such a POMDP can be reduced to the equivalent model of an information-state MDP. See the discussion in [5, Ch. 2]. This supplements a simulation-based algorithm by Baxter and Bartlett [2] for solving average reward POMDPs via gradient-based direct policy search. Finally, the assumption on using the same random number associated with the reward and transition functions can be relaxed. The value function then needs to be redefined and the sampling process of the algorithm needs to be modified to handle the different sources of the randomness for the reward and transition functions.

APPENDIX I
PROOF OF THEOREM 3.1

Define $\Lambda_D(\tau) := (e^{D\tau} - 1 - \tau D)/D^2$, and let $\tau_{\max}$ be a constant satisfying $\tau_{\max} \neq 0$ and $1 + (D + \epsilon)\tau_{\max} - e^{D\tau_{\max}} = 0$. Let $Y_k = \sum_{i=1}^{k}(X_i - \mu)$. It is easy to see that the sequence $\{Y_k\}$ forms a martingale w.r.t $\{\mathcal{F}_k\}$, where $\mathcal{F}_j$ is the σ-field generated by $\{Y_1, \ldots, Y_j\}$. Therefore, for any $\tau > 0$, $P(M^{-1}\sum_{i=1}^{M} X_i - \mu \geq \epsilon, M \geq n) = P(Y_M \geq M\epsilon, M \geq n) = P(\tau Y_M - \Lambda_D(\tau)\langle Y\rangle_M \geq \tau M\epsilon - \Lambda_D(\tau)\langle Y\rangle_M, M \geq n)$, where $\langle Y\rangle_n = \sum_{j=1}^{n} E[(\Delta Y_j)^2|\mathcal{F}_{j-1}], \Delta Y_j = Y_j - Y_{j-1}$.

Now, for any $\tau \in (0, \tau_{\max})$, and for any $n_1 \geq n_0$, where $n_0, n_1 \in \mathcal{Z}^+$, we have $\tau(n_1 - n_0)\epsilon \geq ((e^{D\tau} - 1 - \tau D)/D^2)(n_1 - n_0)D^2 \geq \Lambda_D(\tau)[\sum_{j=1}^{n_1} E[(\Delta Y_j)^2|\mathcal{F}_{j-1}] - \sum_{j=1}^{n_0} E[(\Delta Y_j)^2|\mathcal{F}_{j-1}]]$, which implies that $\tau n_1\epsilon - \Lambda_D(\tau)\langle Y\rangle_{n_1} \geq \tau n_0\epsilon - \Lambda_D(\tau)\langle Y\rangle_{n_0}, \forall \tau \in (0, \tau_{\max})$. Thus, for all $\tau \in (0, \tau_{\max})$

$$\begin{aligned} &P\left(\frac{1}{M}\sum_{i=1}^{M} X_i - \mu \geq \epsilon, M \geq n\right) \\ &\leq P(\tau Y_M - \Lambda_D(\tau)\langle Y\rangle_M \geq \tau n\epsilon - \Lambda_D(\tau)\langle Y\rangle_n, M \geq n) \\ &\leq P(\tau Y_M - \Lambda_D(\tau)\langle Y\rangle_M \geq \tau n\epsilon - \Lambda_D(\tau) n D^2, M \geq n) \\ &= P(e^{\tau Y_M - \Lambda_D(\tau)\langle Y\rangle_M} \geq e^{\tau n\epsilon - n\Lambda_D(\tau)D^2}, M \geq n). \end{aligned} \tag{6}$$

It can be shown that (cf., e.g., [15, Lemma 1, p. 505]) the sequence $\{Z_t(\tau) = e^{\tau Y_t - \Lambda_D(\tau)\langle Y\rangle_t}, t \geq 1\}$ with $Z_0(\tau) = 1$ forms a nonnegative supermartingale. It follows that

$$\begin{aligned} (6) &\leq P(e^{\tau Y_M - \Lambda_D(\tau)\langle Y\rangle_M} \geq e^{\tau n\epsilon - n\Lambda_D(\tau)D^2}) \\ &\leq P\left(\sup_{0 \leq t \leq L} Z_t(\tau) \geq e^{\tau n\epsilon - n\Lambda_D(\tau)D^2}\right) \\ &\leq \frac{E[Z_0(\tau)]}{e^{\tau n\epsilon - n\Lambda_D(\tau)D^2}} = e^{-n(\tau\epsilon - \Lambda_D(\tau)D^2)} \end{aligned}$$

where the last inequality follows from the maximal inequality for supermartingales [15]. By using a similar argument, we can also show that

$$P\left(\frac{1}{M}\sum_{i=1}^{M} X_i - \mu \leq -\epsilon, M \geq n\right) \leq e^{-n(\tau\epsilon - \Lambda_D(\tau)D^2)}. \tag{7}$$

Thus, by combining (6) and (7), we have

$$P\left(\left|\frac{1}{M}\sum_{i=1}^{M} X_i - \mu\right| \geq \epsilon, M \geq n\right) \leq 2e^{-n(\tau\epsilon - \Lambda_D(\tau)D^2)}. \tag{8}$$

Finally, we optimize the right-hand side of (8) over τ. It is easy to verify that the optimal τ^* is given by $\tau^* = D^{-1}\ln((D+\epsilon)/D) \in (0, \tau_{\max})$ and $\tau^*\epsilon - \Lambda_D(\tau^*)D^2 = ((D+\epsilon)/D)\ln((D+\epsilon)/D) - \epsilon/D > 0$. Hence, Theorem 3.1 follows.

Appendix II
Proof of Lemma 3.2

For any action $a \in A$, let $I_j(a)$ be the iteration at which action a is chosen for the jth time, let $\hat{Q}_{i,k}^{K_i}(x,a)$ be the current estimate of $Q_i^*(x,a)$ at the kth iteration, and let $N_a^{i,k}(x)$ be the number of times action a is sampled up to the kth iteration at x, i.e., $N_a^{i,k}(x) = \sum_{j=0}^{k} I\{a(j) = a\}$. By RASA, the estimation $\hat{Q}_{i,k}^{K_i}(x,a)$ is given by [cf. (4)] $\hat{Q}_{i,k}^{K_i}(x,a) = N_a^{i,k}(x)^{-1}\sum_{j=1}^{N_a^{i,k}(x)}(R(x,a,w_{I_j(a)}) + V_{i+1}^*(f(x,a,w_{I_j(a)})))$. Since the sequence of random variables $\{w_{I_j(a)}, j \geq 1\}$ are i.i.d., a straightforward application of Theorem 3.1 yields (9) shown at the bottom of the page. Define the events $\mathcal{A}_k = \{|\hat{Q}_{i,k}^{K_i}(x,a) - Q_i^*(x,a)| \geq \epsilon\}$ and $\mathcal{B}_k = \{N_a^{i,k}(x) \geq N\}$. By the law of total probability, $P(\mathcal{A}_k) = P(\mathcal{A}_k \cap \mathcal{B}_k) + P(\mathcal{A}_k|\mathcal{B}_k^c)P(\mathcal{B}_k^c) \leq P(\mathcal{A}_k \cap \mathcal{B}_k) + P(\mathcal{B}_k^c)$. Taking

$$N = \left\lceil \frac{R_{\max}H\ln(4/\delta)}{(R_{\max}H+\epsilon)\ln((R_{\max}H+\epsilon)/R_{\max}H) - \epsilon} \right\rceil$$

we get from (9) that $P(\mathcal{A}_k \cap \mathcal{B}_k) \leq \delta/2$. On the other hand, by Lemma 3.1, $P(\mathcal{B}_k^c) = P(N_a^{i,k}(x) < N) < \delta/2$ for $k > \bar{\lambda}(N, \delta/2)$ and $0 < \mu_i < 1 - 2^{-1/\bar{\lambda}(N,\delta/2)}$. Therefore, $P(\mathcal{A}_{K_i}) = P(|\hat{Q}_i^{K_i}(x,a) - Q_i^*(x,a)| \geq \epsilon) < \delta$ for $K_i > \lambda(\epsilon,\delta)$ and $0 < \mu_i < \mu_i^*(\epsilon,\delta)$, where

$$\lambda(\epsilon,\delta) = \left\lceil \frac{2M_{\epsilon,\delta}}{\ln l}\ln\left[\frac{lM_{\epsilon,\delta}}{\ln l}\left(\frac{2M_{\epsilon,\delta}}{\delta}\right)^{\frac{1}{M_{\epsilon,\delta}}}\right]\right\rceil$$

$$M_{\epsilon,\delta} = \max\left\{6, \left\lceil \frac{R_{\max}H\ln(4/\delta)}{(R_{\max}H+\epsilon)\ln((R_{\max}H+\epsilon)/R_{\max}H) - \epsilon}\right\rceil\right\}$$

and $\mu_i^*(\epsilon,\delta) = 1 - 2^{-1/K_i} < 1 - 2^{-1/\lambda(\epsilon,\delta)}$. Since $a \in A$ is arbitrary, the proof is complete.

Appendix III
Proof of Lemma 3.3

Let X_s^i be the set of sampled states in X by the algorithm at stage i. Suppose for a moment that for all $x \in X_s^{i+1}$, with some K_{i+1}, μ_{i+1}, and a given $\delta_{i+1} \in (0,1)$

$$P\left(\left|\hat{V}_{i+1}^{K_{i+1}}(x) - V_{i+1}^*(x)\right| > \frac{\theta}{2^{i+2}}\right) < \delta_{i+1}. \tag{10}$$

Consider for $x \in X_s^i$, $\tilde{Q}_i^{K_i}(x,a) = N_a^i(x)^{-1}\sum_{j=1}^{N_a^i(x)} R(x,a,w_j^a) + V_{i+1}^*(f(x,a,w_j^a))$, where $\{w_j^a\}, j = 1,\ldots,N_a^i(x)$ refers to the sampled random number sequence for the sample execution of the action a in the algorithm. We have that for any sampled $x \in X_s^i$ at stage i

$$\hat{Q}_i^{K_i}(x,a) - \tilde{Q}_i^{K_i}(x,a) = \frac{1}{N_a^i(x)} \times \sum_{j=1}^{N_a^i(x)}\left(\hat{V}_{i+1}^{K_{i+1}}\left(f\left(x,a,w_j^a\right)\right) - V_{i+1}^*(f(x,a,w_j^a))\right).$$

Then, under the supposition that (10) holds, for all $a \in A$ at any sampled $x \in X_s^i$ at stage i

$$P\left(\left|\hat{Q}_i^{K_i}(x,a) - \tilde{Q}_i^{K_i}(x,a)\right| \leq \frac{\theta}{2^{i+2}}\right) \geq (1-\delta_{i+1})^{N_a^i(x)} \geq (1-\delta_{i+1})^{K_{i+1}}. \tag{11}$$

This is because if for all w_j^a's, $j = 1,\ldots,N_a^i(x)$, $|V_{i+1}^{K_{i+1}}(f(x,a,w_j^a)) - V_{i+1}^*(f(x,a,w_j^a))| \leq \epsilon$ for $\epsilon > 0$, then $N_a^i(x)^{-1}\sum_{j=1}^{N_a^i(x)}|V_{i+1}^{K_{i+1}}(f(x,a,w_j^a)) - V_{i+1}^*(f(x,a,w_j^a))| \leq \epsilon$, which further implies

$$N_a^i(x)^{-1}\left|\sum_{j=1}^{N_a^i(x)} V_{i+1}^{K_{i+1}}\left(f\left(x,a,w_j^a\right)\right) - V_{i+1}^*\left(f\left(x,a,w_j^a\right)\right)\right| \leq \epsilon$$

and, therefore

$$P\left(\left|\hat{Q}_i^{K_i}(x,a) - \tilde{Q}_i^{K_i}(x,a)\right| \leq \epsilon\right) \geq \prod_{j=1}^{N_a^i(x)} P\left(\left|V_{i+1}^{K_{i+1}}(f(x,a,w_j^a)) - V_{i+1}^*(f(x,a,w_j^a))\right| \leq \epsilon\right).$$

From Lemma 3.2, for all $a \in A$, with $K_i > \lambda(\theta/2^{i+2}, \delta_i)$ and $\mu_i \in (0, 1-2^{-1/K_i})$ for $\delta_i \in (0,1)$

$$P\left(\left|\tilde{Q}_i^{K_i}(x,a) - Q_i^*(x,a)\right| > \frac{\theta}{2^{i+2}}\right) < \delta_i, \qquad x \in X_s^i. \tag{12}$$

Combining (11) and (12), under the supposition of (10)

$$P\left(\left|\hat{Q}_i^{K_i}(x,a) - Q_i^*(x,a)\right| \leq \frac{\theta}{2^{i+2}} + \frac{\theta}{2^{i+2}}\right) \geq (1-\delta_i)(1-\delta_{i+1})^{K_{i+1}}, \qquad x \in X_s^i.$$

This implies that at the **Exit** step

$$P\left(\max_{a\in A}\left|\hat{Q}_i^{K_i}(x,a) - Q_i^*(x,a)\right| < \frac{\theta}{2}\right) \geq (1-\delta_i)(1-\delta_{i+1})^{K_{i+1}}, \qquad x \in X_s^i. \tag{13}$$

From the definition of θ, if $\max_{a\in A}|\hat{Q}_i^{K_i}(x,a) - Q_i^*(x,a)| < \theta/2$, $\hat{Q}_i^{K_i}(x,a^*) > \hat{Q}_i^{K_i}(x,a)$ for all $a \neq a^*$ with $a^* = \arg\max_{a\in A} Q_i^*(x,a)$ (refer the proof of Theorem 3.1 in [11]). Therefore, by the definition of $\hat{V}_i^{K_i}(x)$, ($\hat{V}_i^{K_i}(x) = \max_{a\in A}\hat{Q}_i^{K_i}(x,a) = \hat{Q}_i^{K_i}(x,a^*)$ and $V_i^*(x) = Q_i^*(x,a^*)$), with our choice of $K_i > \lambda(\theta/2^{i+2}, \delta_i)$ and $\mu_i \in (0, 1-2^{-1/K_i})$, we have that

$$P\left(\left|\hat{V}_i^{K_i}(x) - V_i^*(x)\right| > \frac{\theta}{2^{i+1}}\right) < 1 - (1-\delta_i)(1-\delta_{i+1})^{K_{i+1}}$$

if for all $x \in X_s^{i+1}$, with some K_{i+1}, μ_{i+1}, and a given $\delta_{i+1} \in (0,1)$

$$P\left(\left|\hat{V}_{i+1}^{K_{i+1}}(x) - V_{i+1}^*(x)\right| > \frac{\theta}{2^{i+2}}\right) < \delta_{i+1}.$$

$$P\left(\left|\hat{Q}_{i,k}^{K_i}(x,a) - Q_i^*(x,a)\right| \geq \epsilon, N_a^{i,k}(x) \geq N\right) \leq 2e^{-N((R_{\max}H+\epsilon)/(R_{\max}H)\ln((R_{\max}H+\epsilon)/(R_{\max}H)) - \epsilon/(R_{\max}H))} \tag{9}$$

We now apply an inductive argument. Because $\hat{V}_H^{K_H}(x) = V_H^*(x) = 0, x \in X$, with $K_{H-1} > \lambda(\theta/2^{H+1}, \delta_{H-1}) \geq \lambda(\theta/2^H, \delta_{H-1})$ and $\mu_{H-1} \in (0, 1-2^{-1/K_{H-1}})$

$$P\left(\left|\hat{V}_{H-1}^{K_{H-1}}(x) - V_{H-1}^*(x)\right| > \frac{\theta}{2^H}\right) < \delta_{H-1}, \qquad x \in X_s^{H-1}.$$

It follows that with $K_{H-2} > \lambda(\theta/2^H, \delta_{H-2})$ and $\mu_{H-2} \in (0, 1-2^{-1/K_{H-2}})$

$$P\left(\left|\hat{V}_{H-2}^{K_{H-2}}(x) - V_{H-2}^*(x)\right| > \frac{\theta}{2^{H-1}}\right) < 1-(1-\delta_{H-2})(1-\delta_{H-1})^{K_{H-1}}, \qquad x \in X_s^{H-2}.$$

Continuing this way, we finally have that with $K_0 > \lambda(\frac{\theta}{4}, \delta_0)$ and $\mu_0 \in (0, 1-2^{-\frac{1}{K_0}})$

$$P\left(\left|\hat{V}_0^{K_0}(x_0) - V_0^*(x_0)\right| > \frac{\theta}{2}\right) < 1-(1-\delta_0)(1-\delta_1)^{K_1} \times \cdots \times (1-\delta_{H-1})^{K_1 \cdots K_{H-1}}$$

which completes the proof.

APPENDIX IV
PROOF OF THEOREM 3.2

For the first part, define the event $E'(k) = \{P_{x_0}(k)(a^*) > 1-\epsilon\}$ where $a^* = \arg\max_{a \in A} Q_0^*(x_0, a)$. Let $\lambda(\theta/4, \delta_0) = K$. Then, $P(E'(\kappa+K)) \geq P(E'(\kappa+K)|E(K))P(E(K)), \kappa = 1, 2, \ldots,$ where the event $E(K)$ is given as $\{\max_{a \in A} |\hat{Q}_i^{K_i}(x, a) - Q_i^*(x, a)| < \theta/2\}$ at iteration $k = K$.

By selecting $K_0 > K = \lambda(\theta/4, \delta_0)$ and $K_i > \lambda(\theta/2^{i+2}, \delta_i), i = 1, \ldots, H-1$ and μ_i's for $\delta_i \in (0,1)$, $P(E(K)) \geq \rho$ by Lemma 3.3. We will obtain l such that $P(E'(\kappa+K)|E(K)) = 1$ if $\kappa > l$, proving the statement of the theorem.

From the choice of $K = \lambda(\theta/4, \delta_0)$, at iteration $k > K$, for each nonoptimal action $a \neq a^*$, $P_{x_0}(k)(a)$ is decremented by $(1-\mu_0)$. Therefore, $P_{x_0}(\kappa+K)(a^*) = 1 - \sum_{a \neq a^*} P_{x_0}(K)(a)(1-\mu_0)^\kappa$ and $\sum_{a \neq a^*} P_{x_0}(K)(a)(1-\mu_0)^\kappa < \epsilon$ is satisfied if $\kappa > l = \lceil \ln \epsilon / \ln(1-\mu_0^*) \rceil$.

The second part follows immediately from the proof of Lemma 3.3. We skip the details.

REFERENCES

[1] T. P. I. Ahamed, P. S. N. Rao, and P. S. Sastry, "A reinforcement learning approach to automatic generation control," *Electric Power Syst. Res.*, vol. 63, pp. 9–26, 2002.

[2] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *J. Artif. Intell. Res.*, vol. 15, pp. 319–350, 2001.

[3] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[4] H. S. Chang, M. C. Fu, J. Hu, and S. I. Marcus, "An adaptive sampling algorithm for solving Markov decision processes," *Operat. Res.*, vol. 53, no. 1, pp. 126–139, 2005.

[5] H. S. Chang, M. C. Fu, J. Hu, and S. I. Marcus, *Simulation-Based Algorithms for Markov Decision Processes*. London, U.K.: Springer-Verlag, 2007.

[6] H. S. Chang, M. C. Fu, J. Hu, and S. I. Marcus, "An asymptotically efficient simulation-based algorithm for finite horizon stochastic dynamic programming," *IEEE Trans. Autom. Control*, vol. 52, no. 1, pp. 89–94, Jan. 2007.

[7] O. Hernácndez-Lerma, *Adaptive Markov Control Processes*. New York: Springer-Verlag, 1989.

[8] R. Jain and P. Varaiya, "Simulation-based uniform value function estimates of Markov decision processes," *SIAM J. Control Optim.*, vol. 45, no. 5, pp. 1633–1656, 2006.

[9] A. S. Poznyak, K. Najim, and E. Gomez-Ramirez, *Self-Learning Control of Finite Markov Chains*. New York: Marcel Dekker, 2000.

[10] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York: Wiley, 1994.

[11] K. Rajaraman and P. S. Sastry, "Finite time analysis of the pursuit algorithm for learning automata," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 4, pp. 590–598, Aug. 1996.

[12] G. Santharam and P. S. Sastry, "A reinforcement learning neural network for adaptive control of Markov chains," *IEEE Trans. Syst, Man, Cybern. A, Syst. Humans*, vol. 27, no. 5, pp. 588–600, Sep. 1997.

[13] P. S. Sastry, "Systems of learning automata-estimator algorithms and applications," Ph.D. dissertation, Dept. Electr. Eng., Indian Inst. Sci., Bangalore, India, Jun. 1985.

[14] P. S. Sastry and M. A. L. Thathachar, *Networks of Learning Automata: Techniques for Online Stochastic Optimization*. New York: Springer-Verlag, 2003.

[15] A. N. Shiryaev, *Probability*, 2nd ed. New York: Springer-Verlag, 1995.

[16] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[17] M. A. L. Thathachar and P. S. Sastry, "Varieties of learning automata: An overview," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 6, pp. 711–722, Dec. 2002.

[18] R. M. Wheeler Jr. and K. S. Narendra, "Decentralized learning in finite Markov chains," *IEEE Trans. Autom. Control*, vol. AC-31, no. 6, pp. 519–526, Jun. 1986.