Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Recursive Learning Automata for Control of Partially Observable Markov Decision Processes

Hyeong Soo Chang*
Department of Computer Science and Engineering
Sogang University
Seoul, Korea
hschang@sogang.ac.kr

Michael C. Fu**
The Robert H. Smith School of Business
University of Maryland
College Park, MD, USA
mfu@rhsmith.umd.edu

Steven I. Marcus**
Department of Electrical and Computer Engineering
University of Maryland
College Park, MD, USA
marcus@eng.umd.edu

***Abstract*— This paper presents a sampling algorithm, called "Recursive Automata Sampling Algorithm (RASA)," for control of finite horizon information-state Markov decision processes (MDPs), the equivalent model of partially observable MDPs. RASA extends in a recursive manner the Pursuit algorithm designed with learning automata by Rajaraman and Sastry for solving stochastic optimization problems. Based on the finite-time analysis of the Pursuit algorithm, we analyze the finite-time behavior of RASA, providing a bound on the probability that a given initial state takes the optimal action, and a bound on the probability that the difference between the optimal value and the estimate of it exceeds a given error. We also discuss how to apply RASA in the direct context of POMDPs and how to incorporate heuristic knowledge into RASA for on-line control.**

I. Introduction

Consider a discrete-time dynamic system with a finite horizon $H < \infty$:

$$x_{i+1} = f(x_i, a_i, w_i) \text{ for } i = 0, 1, 2, ..., H-1,$$

where x_i is a random variable ranging over an 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 a next-state function. The system model covers the dynamics of partially observable MDPs (POMDPs) with finite state, action, and observation spaces, as such a POMDP can be reduced to the equivalent model of *information-state* MDP [1], where the state space is the set of all possible probability distributions over the state space of the corresponding POMDP (see Section V).

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

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

where $x \in X, w = (w_i, w_{i+1}, ..., w_{H-1}), w_j \sim U(0,1), j = i, ..., H-1$, with a bounded nonnegative reward function $R : X \times A \times [0,1] \to \mathcal{R}^+$, $V_H^*(x) = 0$ 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, we wish to compute $V_0^*(x_0)$ and obtain an optimal policy $\pi^* \in \Pi$ that achieves $V_0^*(x_0), x_0 \in X$. 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 paper presents a sampling algorithm, called "Recursive Automata Sampling Algorithm (RASA)," for estimating $V_0^*(x_0)$ and obtaining an approximately optimal policy. RASA extends in a recursive manner (for sequential decisions) the Pursuit algorithm by Rajaraman and Sastry [11] in the context of solving (information-state) MDPs (we will also discuss applying RASA in the direct context of POMDPs). The algorithm complexity is independent of the state space size.

The Pursuit algorithm is based on well-known learning automata [15] for solving (non-sequential) 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 obtained from the current estimates of the average rewards of taking each action (see [11] for a detailed discussion).

As learning automata 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.

Based on the finite-time analysis of the Pursuit algorithm,

*H. S. Chang is also affiliated with Program of Integrated Biotechnology at Sogang University. This work was supported by the Ministry of Science and Technology 21st Century Frontier Program: Intelligent Robot Project in 2005.

**This work was supported in part by the National Science Foundation under Grant DMI-0323220, and in part by the Air Force Office Office of Scientific Research under Grant FA95500410210.

0-7803-9568-9/05/$20.00 ©2005 IEEE

we analyze the finite-time behavior of RASA, providing a bound on the probability that the initial state at stage 0 takes the optimal action in terms of sampling parameters of RASA and a bound on the probability that the difference between the estimate of $V_0^*(x_0)$ and $V_0^*(x_0)$ exceeds a given error.

This paper is organized as follows. In Section II, we discuss some related works. We present RASA in Section III and analyze it in Section IV. In Section V, we discuss how to apply RASA in the context of POMDPs and how to incorporate heuristic knowledge to RASA for on-line control. We conclude our paper in Section VI.

II. Related Works

The book by Poznyak et al. [9] provides a good survey of works on using learning automata for solving controlled (ergodic) Markov chains in model-free reinforcement learning (RL) framework for a loss function defined on the chains. Each automaton in learning automata 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 [16] consider controlling ergodic Markov chains for the infinite horizon average reward within the similar RL framework. For all learning automata approaches so far, to the authors' best knowledge, the number of the automata grows with the size of the state space and thus the correspondingly updating procedure becomes cumbersome except in Wheeler and Narendra's algorithm, where only the currently visited automaton (state) updates relevant functions.

Santharam and Sastry [12] provide an adaptive method of solving 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 transition probabilities. As in [16], 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 exponentially with the sizes of the state and action spaces.

Jain and Varaiya [4] 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 over the entire set of feasible policies.

It is well-known that the optimal value function in a POMDP has a piecewise linear and convex structure. Exploiting the structure, at each iteration of value iteration [13] the complete information-state simplex is searched for a minimal set of information-states that generate the necessary set of hyperplanes, or vectors, over the information-state space, where the vectors parametrize the next value function. This search in general requires solving linear programming problems, making application of value iteration intractable in many cases. Recent works on approximating value iteration focus on computing approximate value function only for a fixed/dynamically growing sampled set of information-states, termed as points. The crucial issue of these approaches is how to choose the points for approximation. The grid-based approaches typically suffer from expensive interpolation and/or scalability problem [2] [7] [17]. The works in [5] [8] [14] consider obtaining the points by actual simulation of POMDPs by sampling random actions and observations at each time step of value iterations. These approaches are different from RASA in that firstly, RASA is not in a form of value iteration (i.e., RASA does not iteratively generate a new value function over sampled information-states from a old value function); rather it builds a sampled tree of information-states and directly estimates the value of the state at the root of the tree in a bottom-up fashion; and secondly, RASA incorporates an adaptive sampling mechanism for selecting which action to sample. Thirdly, it avoids the issues regarding grid-based value-function representation.

The properties of constructing a sampled tree in a recursive manner to estimate the optimal value $V_0^*(x_0)$ at an initial state x_0 and making the use of an adaptive action sampling while constructing the tree is similar to the adaptive multi-stage sampling (AMS) algorithm presented by Chang et al. [3]. The adaptive sampling in AMS is based on the exploration-exploitation process of multi-armed bandit from regret analysis. In contrast, RASA's sampling is based on the probability estimate for the optimal action. The analysis of AMS is given in terms of the expected bias. Here we provide a probability bound for the performance analysis of RASA.

III. Algorithm

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

$$\begin{aligned} V_i^*(x) &= \max_{a \in A} Q_i^*(x,a), \text{ where} \\ Q_i^*(x,a) &= E_w[R(x,a,w) + V_{i+1}^*(f(x,a,w))], \quad (1) \end{aligned}$$

where $w \sim U(0,1)$ and $V_H^*(x) = 0, x \in X$.

We provide below a high-level description of the recursive automata sampling algorithm (RASA) to estimate $V_0^*(x_0)$ for a given initial state x_0 via the relationship given in Equation (1). The inputs to RASA are a state $x \in X$, sampling parameters $K_i > 0$, $\mu_i \in (0,1)$, and stage i, and the output of RASA is $\hat{V}_i^{K_i}(x)$, the estimate of $V_i^*(x)$ where $\hat{V}_H^{K_H}(x) = 0$ for any K_H and $x \in X$. Whenever we encounter $\hat{V}_{i'}^{K_{i'}}(y)$ for a state $y \in X$ and stage i' in the **Loop** portion of RASA, we need to call RASA recursively (at Equation (3)). The initial call to RASA is done with $i = 0$, the initial state x_0, K_0, and μ_0 and every sampling is done independently of the previously done 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 actions and random numbers (an

action and a random number together corresponding to an edge in the tree) K_0 times independently w.r.t. the current probability distribution $P_{x_0}(k)$ and $U(0,1)$, respectively, at each iteration $k = 0, ..., K_0 - 1$. If action $a(k) \in A$ is sampled, the count variable $N^0_{a(k)}(x_0)$ is incremented. By the sampled action $a(k)$ and the random number w_k, the automaton obtains an environment response

$$R(x_0, a(k), w_k) + \hat{V}_1^{K_1}(f(x_0, a(k), w_k)). \quad (2)$$

Then by averaging the responses obtained for each action $a \in A$, the x_0-automaton computes a sample mean of $Q^*_0(x_0, a), a \in A$:

$$\hat{Q}_0^{K_0}(x_0, a) = \frac{1}{N^0_a(x_0)} \sum_{j=1}^{N^0_a(x_0)} R(x_0, a, w^a_j) + \hat{V}_1^{K_1}(f(x_0, a, w^a_j)),$$

where $\{w^a_j\}, j = 1, ..., N^0_a(x_0)$ are the random numbers generated when the action a was sampled. Note that $\sum_{a' \in A} N^0_{a'}(x_0) = K_0$. The x_0-automaton then obtains the current optimal action that achieves the current best value $\max_{a \in A} \hat{Q}_0^{K_0}(x_0, a)$ (ties are resolved arbitrarily), updates P_{x_0} as in Equation (6) and (7), and estimates the optimal value $V^*_0(x_0)$ by $\hat{V}_0^{K_0}(x_0) = \max_{a \in A} \hat{Q}_0^{K_0}(x_0, a)$. The probability distribution $P_{x_0}(k)$ is updated in the direction of the current estimate of the optimal action $\hat{a}^* = \arg\max_{a' \in A} \hat{Q}_0^{K_0}(x_0, a')$ at each iteration k. 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].

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 $\hat{V}_1^{K_1}(f(x, a(k), w_k))$ compose the environment responses for the x_0-automaton.

Recursive Automata Sampling Algorithm (RASA)

- **Input:** state $x \in X$, $K_i > 0$, $\mu_i \in (0,1)$, stage i. **Output:** $\hat{V}_i^{K_i}(x)$.
- **Initialization:** Set $P_x(0)(a) = 1/|A|, N^i_a(x) = 0, M_i^{K_i}(x,a) = 0, a \in A$ and $k = 0$.
- **Loop:** $(i \neq H)$
  - Sample $a(k) \sim P_x(k)$, $w_k \sim U(0,1)$.
  - Update $\hat{Q}_i^{K_i}(x,a)$ only for $a = a(k)$ such that

$$M_i^{K_i}(x,a) \leftarrow M_i^{K_i}(x,a) + R(x,a,w_k) + \hat{V}_{i+1}^{K_{i+1}}(f(x,a,w_k)); \quad (3)$$

$$N^i_a(x) \leftarrow N^i_a(x) + 1; \quad (4)$$

$$\hat{Q}_i^{K_i}(x,a) \leftarrow \frac{M_i^{K_i}(x,a)}{N^i_a(x)}; \quad (5)$$

  - For $\hat{a}^* = \arg\max_{a' \in A} \hat{Q}_i^{K_i}(x,a')$,

$$P_x(k+1)(\hat{a}^*) \leftarrow (1-\mu_i)P_x(k)(\hat{a}^*) + \mu_i; \quad (6)$$

  and for all $a \neq \hat{a}^*$,

$$P_x(k+1)(a) \leftarrow (1-\mu_i)P_x(k)(a). \quad (7)$$

  - $k \leftarrow k+1$. If $k = K_i$, then exit **Loop**.
- **Exit:** Return $\hat{V}_i^{K_i}(x) = \max_{a \in A} \hat{Q}_i^{K_i}(x,a)$ if $i \neq H$, and 0 otherwise.

The running time complexity of the RASA algorithm is $O(K^H)$ with $K = \max_i K_i$, independent of the state space size. (For some performance guarantee, the value K depends on the size of the action space. See the next section.)

IV. Analysis of RASA

All of the estimated $\hat{V}^{K_i}$ and $\hat{Q}^{K_i}$-values in the current section refer to the values at the **Exit** in RASA. The following lemma 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 recursive call is no longer done), which is proven by Rajaraman and Sastry [11, Lemma 3.2] in their context.

Lemma 1: Consider the nonrecursive algorithm RASA' obtained by replacing Equation (3) by

$$M_i^{K_i}(x,a) \leftarrow M_i^{K_i}(x,a) + R(x,a,w_k) + V^*_{i+1}(f(x,a,w_k)). \quad (8)$$

Assume $K_i > \lambda(\epsilon, \delta)$ and $0 < \mu_i < \mu^*_i(\epsilon, \delta)$ where

$$\lambda(\epsilon, \delta) = \left\lceil \frac{2M_{\epsilon,\delta}}{\ln l} \ln \left\lceil \frac{lM_{\epsilon,\delta}}{\ln l} \left(\frac{2M_{\epsilon,\delta}}{\delta} \right)^{\frac{1}{M_{\epsilon,\delta}}} \right\rceil \right\rceil \quad (9)$$

with $M_{\epsilon,\delta} = \max\{6, \lceil \frac{R^2_{\max} H^2}{2\epsilon^2} \ln \frac{4}{\delta} \rceil\}$, $l = 2|A|/(2|A|-1)$, and $\mu^*_i(\epsilon,\delta) = 1 - 2^{-\frac{1}{K_i}}$. Consider a fixed i, $x \in X$, $\epsilon > 0$, and $\delta \in (0,1)$. Then, for all $a \in A$ at the **Exit** step

$$\Pr\{|\hat{Q}_i^{K_i}(x,a) - Q^*_i(x,a)| > \epsilon\} < \delta.$$

We now put an assumption *for the purpose of the analysis*. The assumption states that at each stage, the optimal action is unique at each state. In other words, the given MDP has a unique optimal nonstationary policy. We will give a remark on this at the end of this section.

Assumption 1: For all $x \in X$ and $i = 0, 1, ..., H-1$, $\theta_i(x) := Q^*_i(x, a^*) - \max_{a \neq a^*} Q^*_i(x,a) > 0$ where $V^*_i(x) = Q^*_i(x, a^*)$. We let $\inf_{x \in X, i = 0, ..., H-1} \theta_i(x) = \theta$.

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

$$\delta := (1-\delta_0) \prod_{i=1}^{H-1} (1-\delta_i)^{\prod_{j=1}^{i} K_j}$$

.

Lemma 2: Assume that Assumption 1 holds. Select $K_i > \lambda(\frac{\theta}{2^{i+2}}, \delta_i), i = 0, ..., H-1$ and $0 < \mu_i < \mu^*_i = 1 - 2^{-\frac{1}{K_i}}$ for a given $\delta_i \in (0,1)$. Then under RASA,

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

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

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

Consider for $x \in X_s^i$,

$$\tilde{Q}_i^{K_i}(x,a) = \frac{1}{N_a^i(x)} \sum_{j=1}^{N_a^i(x)} R(x,a,w_j^a) + V_{i+1}^*(f(x,a,w_j^a)),$$

where $\{w_j^a\}, j = 1, ..., N_a^i(x)$ refers to the sampled random number sequence for the sample execution of the action a in RASA. We have that for any sampled $x \in X_s^i$ at stage i,

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

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

$$\Pr\left\{\left|\hat{Q}_i^{K_i}(x,a) - \tilde{Q}_i^{K_i}(x,a)\right| \le \frac{\theta}{2^{i+2}}\right\}$$
$$\ge (1-\delta_{i+1})^{N_a^i(x)} \ge (1-\delta_{i+1})^{K_{i+1}}. \quad (10)$$

This is because if for all of the next states sampled by taking a, $|V_{i+1}^{K_{i+1}}(f(x,a,w_j^a)) - V_{i+1}^*(f(x,a,w_j^a))| \le \epsilon$ for $\epsilon > 0$, then

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

which further implies

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

and therefore

$$\Pr\{|\hat{Q}_i^{K_i}(x,a) - \tilde{Q}_i^{K_i}(x,a)| \le \epsilon\} \ge$$
$$\prod_{j=1}^{N_a^i(x)} \Pr\{|V_{i+1}^{K_{i+1}}(f(x,a,w_j^a)) - V_{i+1}^*(f(x,a,w_j^a))| \le \epsilon\}.$$

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

$$\Pr\left\{\left|\tilde{Q}_i^{K_i}(x,a) - Q_i^*(x,a)\right| > \frac{\theta}{2^{i+2}}\right\} < \delta_i, x \in X_s^i. \quad (11)$$

Combining Equation (10) and (11),

$$\Pr\left\{\left|\hat{Q}_i^{K_i}(x,a) - Q_i^*(x,a)\right| \le \frac{\theta}{2^{i+2}} + \frac{\theta}{2^{i+2}}\right\}$$
$$\ge (1-\delta_i)(1-\delta_{i+1})^{K_{i+1}},$$

and this yields that under the supposition, for any $x \in X_s^i$,

$$\Pr\{|\hat{Q}_i^{K_i}(x,a) - Q_i^*(x,a)| \le \frac{\theta}{2^{i+1}}\} \ge (1-\delta_i)(1-\delta_{i+1})^{K_{i+1}}.$$

Define the event

$$E(K_i) = \left\{\sup_{j \ge K_i} \max_{a \in A} \left|\hat{Q}_i^j(x,a) - Q_i^*(x,a)\right| < \frac{\theta}{2}\right\}. \quad (12)$$

If the event $E(K_i)$ occurs at the **Exit** step, then from the definition of θ and the event $E(K_i)$, $\hat{Q}_i^{K_i}(x,a^*) > \hat{Q}_i^{K_i}(x,a)$ for all $a \ne a^*$ with $a^* = \arg\max_{a \in A} Q_i^*(x,a)$ (Refer the proof of Theorem 3.1 in [11]). Because from 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 $K_i > \lambda(\frac{\theta}{2^{i+2}}, \delta_i)$,

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

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

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

We now apply an inductive argument: because $\hat{V}_H^{K_H}(x) = V_H^*(x) = 0, x \in X$, with $K_{H-1} > \lambda(\frac{\theta}{2^{H+1}}, \delta_{H-1}) \ge \lambda(\frac{\theta}{2^H}, \delta_{H-1})$,

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

It follows that with $K_{H-2} > \lambda(\frac{\theta}{2^H}, \delta_{H-2})$,

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

and further follows that with $K_{H-3} > \lambda(\frac{\theta}{2^{H-1}}, \delta_{H-3})$

$$\Pr\left\{\left|\hat{V}_{H-3}^{K_{H-3}}(x) - V_{H-3}^*(x)\right| > \frac{\theta}{2^{H-2}}\right\}$$
$$< 1-(1-\delta_{H-3})(1-\delta_{H-2})^{K_{H-2}}$$
$$\times(1-\delta_{H-1})^{K_{H-2}K_{H-1}}, x \in X_s^{H-3}.$$

Continuing this way, we have that

$$\Pr\left\{\left|\hat{V}_1^{K_1}(x) - V_1^*(x)\right| > \frac{\theta}{2^2}\right\}$$
$$< 1-(1-\delta_1)(1-\delta_2)^{K_2}(1-\delta_3)^{K_2K_3}$$
$$\times \cdots \times (1-\delta_{H-1})^{K_2 \cdots K_{H-1}}, x \in X_s^1$$

and for $K_0 > \lambda(\frac{\theta}{4}, \delta_0)$

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

which completes the proof. ■

Theorem 1: Assume that Assumption 1 holds. Given $\delta_i \in (0,1), i = 0, ..., H-1$, select $K_i > \lambda(\frac{\theta}{2^{i+2}}, \delta_i)$ and $0 < \mu_i < \mu_i^* = 1 - 2^{-\frac{1}{K_i}}, i = 1, ..., H-1$. Then under RASA, for all $\epsilon \in (0,1)$,

$$\Pr\{P_{x_0}(K_0)(a^*) > 1-\epsilon\} > 1-\delta,$$

where $a^* = \arg\max_{a \in A} Q_0^*(x_0,a)$ and $K_0 > K_0^*$, where

$$K_0^* = \kappa + \left\lceil \frac{\ln \frac{1}{\epsilon}}{\ln \frac{1}{1-\mu_0^*}} \right\rceil$$

with $\kappa > \lambda(\frac{\theta}{4}, \delta_0)$ and $0 < \mu_0 < \mu_0^* = 1 - 2^{-\frac{1}{\kappa}}$.

Proof: Define the event $E'(K_0) = \{P_{x_0}(K_0)(a^*) > 1 - \epsilon\}$ where $a^* = \arg\max_{a \in A} Q_0^*(x_0, a)$. Then,

$$\Pr\{P_{x_0}(K_0^*)(a^*) > 1 - \epsilon\} = \Pr(E'(K_0^*)) \geq \Pr(E'(K_0^*)|E(\kappa)) \Pr(E(\kappa)),$$

where the event E is given in Equation (12) with $i = 0$.

From the selection of $\kappa > \lambda(\frac{\theta}{4}, \delta_0)$ with $K_i > \lambda(\frac{\theta}{2^{i+2}}, \delta_i), i = 1, ..., H-1$ and μ_i's for $\delta_i \in (0, 1)$, $\Pr(E(\kappa)) \geq 1 - \delta$ by Lemma 2. With reasoning similar to that in the proof of Theorem 3.1 in [11], $\Pr(E'(l + \kappa)|E(\kappa)) = 1$ if $l + \kappa > \lambda(\frac{\theta}{2}, \delta_0) + \lceil \frac{\ln \frac{1}{\epsilon}}{\ln \frac{1}{1-\mu_0^*}} \rceil$. ■

Based on the proof of Lemma 2, the following result can be stated.

Theorem 2: Assume that Assumption 1 holds. Given $\delta_i \in (0, 1), i = 0, ..., H-1$ and $\epsilon \in (0, \theta]$, select $K_i > \lambda(\frac{\epsilon}{2^{i+2}}, \delta_i), 0 < \mu_i < \mu_i^* = 1 - 2^{-\frac{1}{K_i}}, i = 0, ..., H-1$. Then under RASA,

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

As we can see from the statements of Lemma 2, Theorems 1 and 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 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 RASA will be (the smaller sampling complexity). Therefore, in the actual implementation of RASA, if multiple actions' performances 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 [11].

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 1 and 2 are obtained with coarse estimation of various parameters/terms. For example, we used the worst case values of $\theta_i(x), x \in X, i = 0, ..., H-1$ and $(R_{\max} H)^2$ for bounding $\sup_{x \in X} V_i^*(x), i = 0, ..., H-1$, and used conservative bounds in Equation (10) 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.

V. Discussion

A. Applying RASA to POMDPs

Consider a POMDP model parameterized as follows. X is a finite set of states, A is a finite set of actions, O is a finite set of observations that provide incomplete state information, and I_0 is the initial information-state, i.e., a probability distribution over X ($I_0(x), x \in X$ denotes the probability of being in state $x \in X$). At stage i, the system is in x_i (where this state information is unknown to the controller; the controller estimates x_i by I_i). The controller takes an action a_i, and the system makes a transition to x_{i+1} by the probability $P(x_{i+1}|x_i, a_i)$ and the controller obtains the reward of $r(x_i, a_i, x_{i+1})$. At stage $i+1$, the controller observes an observation generated with the probability $O(o_{i+1}|x_{i+1}, a_i)$. The controller updates its information-state by

$$I_{i+1}(y) = \eta O(o_{i+1}|y, a_i) \sum_{x \in X} P(y|x, a_i) I_i(x), y \in X$$

where η is the normalizing constant. From this information-state update procedure, we can induce the probability $P(I_{i+1}|I_i, a_i)$ and map this into a next state function $h : X_I \times A \times [0, 1] \rightarrow X_I$, where X_I is the space of all possible information-states. The reward function $R_I : X_I \times A \times [0, 1] \rightarrow \mathcal{R}^+$ is similarly induced. Once the equivalent information-state MDP is built up, RASA can be applied to the information-state MDP.

But we can modify the RASA description in the pseudocode and apply the modified RASA to the unreduced model of the POMDP (Input to RASA is an information-state x in X_I). Simply modify **Loop** as follows (we state only changes):

- **Loop:** $(i \neq H)$
  - Sample $a(k)$ w.r.t. $P_x(k)$ and sample $y \in X$ w.r.t. $I_i \in X_I$.
  - Sample $z \in X$ w.r.t. $P(\cdot|y, a(k))$.
  - Sample o w.r.t. $O(\cdot|z, a(k))$.
  - Obtain the information-state I_{i+1}^k: for $y \in X$,

$$I_{i+1}(k)(y) = \eta O(o|y, a(k)) \sum_{y' \in X} P(y|y', a(k)) I_i(y')$$

  - Update only $\hat{Q}_i^{K_i}(x, a)$ with $a = a(k)$ such that

$$M_i^{K_i}(x, a) \leftarrow M_i^{K_i}(x, a) + \textstyle\sum_{z \in X} r(x, a, z) I_{i+1}(z) + \hat{V}_{i+1}^{K_{i+1}}(I_{i+1}^k) \quad (13)$$

Once armed with an algorithm that estimates the optimal value/policy for finite horizon problems, we can create a nonstationary stochastic policy in an on-line manner in the context of "planning" or receding horizon control [3] [6] for solving infinite horizon problems.

B. Incorporating heuristic knowledge into RASA

Suppose that we wish to control information-state MDPs or POMDPs in an on-line manner and have some heuristic

knowledge on the problem solutions. For example, for the multiclass scheduling problems of stochastically arriving prioritized tasks with deadlines, "earliest deadline first" and "static priority" might be good candidate policies. (Here, the scheduler has no knowledge on the traffic generator's state, yielding a POMDP formulation.) In this case, the decision maker may well wish to incorporate knowledge of those policies into the on-line control of the scheduling system in such a way that for some time, a control to be taken at the current state is generated from the heuristic policies or knowledge while the system is learning/estimating the optimal action (termed as "on-policy learning" in artificial intelligence literature) but in a systematic way.

Let us assume that we use RASA in the context of receding horizon control. We represent our heuristic knowledge as a function $\zeta : X \times A \to [0, 1]$ such that for all $x \in X$, $\sum_{a \in A} \zeta(x, a) = 1$, which estimates the likelihood that at $x \in X$, $a \in A$ is optimal (heuristically estimated). If we know that for sure at some states applying the earliest deadline first policy is optimal, then ζ-function values for these states can be set to one for the corresponding action from earliest deadline first policy.

We then modify RASA by replacing the step in **Loop** containing Equation (6) and (7) with the following:

– For $\hat{a}^* = \arg\max_{a' \in A} \hat{Q}_i^{K_i}(x, a')$,

$$P_x(k+1)(\hat{a}^*) \leftarrow (1 - \mu_i) P_x(k)(\hat{a}^*) + \mu_i \qquad (14)$$

and for all $a \neq \hat{a}^*$,

$$P_x(k+1)(a) \leftarrow (1 - \mu_i) P_x(k)(a). \qquad (15)$$

– Update τ_x and normalize P_x: for all $a \in A$,

$$\tau_x(k+1)(a) \leftarrow \zeta(x, a)^{\beta_{k+1}} P_x(k+1)(a) \qquad (16)$$

$$P_x(k+1)(a) \leftarrow \frac{\tau_x(k+1)(a)}{\sum_{a' \in A} \tau_x(k+1)(a')} \qquad (17)$$

We then control the degree of emphasis on the ζ-value by the tuning parameter β, analogous to cooling the temperature in simulated annealing from high to low values, approaching 0, in Equation (16). We can then extend the result of Lemma 1 in [11] within this modification under the assumption that $\inf_{x \in X, a \in A, k} \zeta(x, a)^{\beta_k} > 0$, and correspondingly Lemma 2 in the present paper, providing the similar finite-time bounds for the modified RASA. Due to space constraints, we defer this discussion to the full version of this paper.

VI. Concluding Remarks

RASA is mainly targetted for *on-line* control whereas grid-based value-iteration approaches are for off-line control. An interesting future research topic would be comparing the two approaches along with other sampling-based algorithms in terms of the sampling complexities and the performances.

References

[1] A. Arapostathis, V. S. Borkar, E. Fernández-Gaucherand, M. K. Ghosh, and S. I. Marcus, "Discrete-time controlled Markov processes with average cost criterion: a survey," *SIAM J. on Control and Optimization*, vol. 31, no. 2, pp. 282–344, 1993.

[2] B. Bonet, "An ϵ-optimal grid-based algorithm for partially observable Markov decision processes," in *Proc. of the 19th Int. Conf. on Machine Learning*, 2002, pp. 51–58.

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

[4] R. Jain and P. Varaiya, "Simulation-based uniform value function estimates of discounted and average-reward MDPs," in *Proc. of the 43rd IEEE Conf. on Decision and Control,*, 2004, pp. 4405–4410.

[5] M. Hauskrecht, "Value function approximations for partially observable Markov decision processes," *J. of Artificial Intelligence Research,* vol. 13, pp. 33–95, 2000.

[6] O. Hernández-Lerma and J. B. Lasserre, "Error bounds for rolling horizon policies in discrete-time Markov control processes," *IEEE Trans. on Automatic Control*, vol. 35, pp. 1118–1124, 1990.

[7] W. S. Lovejoy, "Computationally feasible bounds for partially observed Markov decision processes," *Operations Research*, vol. 39, no. 1, pp. 162–175, 1991.

[8] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: an anytime algorithm for POMDPs," in *Proc. of the Int. Joint Conf. on Artificial Intelligence,* 2003.

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

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

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

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

[13] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable Markov decision processes over a finite horizon," *Operations Research*, vol. 21, pp. 1071–1088, 1973.

[14] M. T. J. Spaan and N. Vlassis, "A point-based POMDP algorithm for robot planning," in *Proc. of the IEEE Conf. on Robotics and Automation,* 2004, pp. 2399–2404.

[15] M. A. L. Thathachar and P. S. Sastry, "Varieties of learning automata: an overview," *IEEE Trans. on Systems, Man, and Cybernetics,* Part B, vol. 32, no. 6, pp. 711–722, 2002.

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

[17] R. Zhou and E. A. Hansen, "An improved grid-based approximation algorithm for POMDPs," in *Proc. of the Int. Joint Conf. on Artificial Intelligence*, 2001.