A Reinforcement Learning Based Algorithm for Finite Horizon Markov Decision Processes

Shalabh Bhatnagar and Mohammed Shahid Abdulla
Department of Computer Science and Automation,
Indian Institute of Science, Bangalore, INDIA.
e-mail:{shalabh,shahid}@csa.iisc.ernet.in

***Abstract*— We develop a simulation based algorithm for finite horizon Markov decision processes with finite state and finite action space. Illustrative numerical experiments with the proposed algorithm are shown for problems in flow control of communication networks and capacity switching in semiconductor fabrication.**

Keywords

Finite horizon Markov decision processes, reinforcement learning, two timescale stochastic approximation, actor-critic algorithms, normalized Hadamard matrices.

I. Introduction

Markov decision processes (MDPs) are a general framework for solving stochastic control problems [1]. Value iteration and policy iteration are two of the classical approaches for solving the Bellman equation for optimality. Whereas value iteration proceeds by recursively iterating over value function estimates starting from a given such estimate, policy iteration does so by iterating over policies and involves updates in two nested loops. The inner loop estimates the value function for a given policy update while the outer loop updates the policy. The former estimates are obtained as solutions to linear systems of equations, known as the Poisson equations, that are most often solved using value iteration type recursions rather than explicit matrix inversion. This is particularly true when the numbers of states and actions are large (the 'curse of dimensionality'). However, in the above classical approaches, one requires complete knowledge of the system model via transition probabilities. Even if these are available, the 'curse of dimensionality' threatens the computational requirements for solving the Bellman equation as these become prohibitive. Motivated by these considerations, research on simulation-based methods that largely go under the rubric of reinforcement learning or neuro-dynamic programming [2] have gathered momentum in recent times. The main idea in these schemes is to simulate transitions instead of directly computing transition probabilities and, in scenarios where the numbers of states and actions are large, use parametric representations of the cost-to-go function and/or policies.

The inner loop of the policy iteration algorithm, for any given policy update, may typically take a long time to converge. In [3], an actor-critic algorithm based on two-timescale stochastic approximation was proposed. A simulation-based analog of policy iteration, the algorithm proceeds using two coupled recursions driven by different step-size schedules or timescales. The policy evaluation step of policy iteration is performed on the faster timescale while the policy improvement step is carried out along the slower one, the advantage being that one need not wait for convergence of the inner-loop before an outer-loop update unlike regular policy iteration. Instead, both recursions are executed in tandem, one after the other, and the optimal policy-value function pair are obtained upon convergence of the algorithm. This idea of two-timescale stochastic approximation is further applied in [4], where parameterizations of both value function (termed 'critic'), and policy (termed 'actor') are considered. As yet another application of the two-timescale method, the simulation-based policy iteration algorithm of [5] performs updates in the space of deterministic stationary policies and not randomized stationary policies (RSPs) as in [3], [4]. While not stationary, the proposed algorithm RPAFA uses randomized policies as well. On the slower timescale, a gradient search using simultaneous perturbation stochastic approximation (SPSA) gradient estimates is performed and convergence to a locally optimal policy is shown. While gradient search on the slower timescale is recommended in [4], no specific form of the gradient estimates is proposed there. The SPSA estimates used in [5] are of the two-measurement form first proposed in [6], while a one-measurement form of SPSA was proposed in [7]. A performance-enhancing modification to this latter algorithm was described in [8] that used deterministic perturbation sequences derived from certain normalized Hadamard matrices. This last form of SPSA is used in the proposed RPAFA algorithm.

The algorithm of [5] is for infinite horizon discounted cost MDPs. Obtaining a solution to the Poisson equation (along the faster timescale) is simpler there as the cost-to-go depends only on the state and is stage-invariant (i.e. stationary). Since we consider the finite horizon setting here, the cost-to-go is now a function of both state and stage. The faster timescale updates now involve T 'stage-wise' coupled stochastic recursions in addition to being 'state-wise' coupled, where T is the planning horizon. Therefore, the resulting system of equations that need to be solved is $T-$fold larger than in [5]. Numerical experiments in [5] are shown over a setting of flow control in communication networks. We consider experiments not only in this setting, but also on another setting involving capacity allocation in semi-conductor fabs. Further, as already pointed out, [5] considers the setting of compact (non-discrete) action sets while we consider a finite action setting in our work.

Reinforcement learning algorithms have generally been developed and studied as infinite horizon MDPs under the discounted cost or the long-run average cost criteria. For instance, approximate DP methods of TD learning [2, §6.3], Q-learning [2, §6.6] and actor-critic

algorithms [4] etc., have been developed in the infinite horizon framework. However, in most real life scenarios, finite horizon decision problems assume utmost significance. For instance, in the design of a manufacturing fab, one requires planning over a finite decision horizon. In a communication network, flow and congestion control problems should realistically be studied only as finite horizon decision making problems, since the amount of time required in clearing congestion and restoring normal traffic flows in the network is of prime concern. Finite-horizon tasks also form natural subproblems in certain kinds of infinite-horizon MDPs, e.g. [9, §2] illustrates this by using models from semiconductor fabrication and communication networks where each transition of the upper level infinite-horizon MDP spawns a finite-horizon MDP at a lower level. Policies in finite horizon problems depend on stage and need not be stationary, thereby contributing in severity to the 'curse of dimensionality'.

We develop in this paper, two-timescale stochastic approximation based actor-critic algorithms for finite horizon MDPs with finite state and finite action sets. Most of the work on developing computationally efficient algorithms for finite horizon problems, however, assumes that model information is known. For instance, in [10], the problem of solving a finite horizon MDP under partial observations is formulated as a nonlinear programming problem and a gradient search based solution methodology is developed. For a similar problem, a solution procedure based on genetic algorithms and mixed integer programming is presented in [11]. In [12], a hierarchical structure using state aggregation is proposed for solving finite horizon problems. In contrast, we assume that information on transition probabilities (or model) of the system is not known, although transitions can be simulated. In [13], three variants of the Q-learning algorithm for the finite horizon problem are developed assuming lack of model information. However, the finite horizon MDP problem is embedded as an infinite horizon MDP either by adding an absorbing state at the terminal stage (or the end of horizon) or a modified MDP is obtained by restarting the process by selecting one of the states at the initial (first) stage of the MDP according to the uniform distribution, once the terminal stage is hit.

Our approach is fundamentally different from that in [13]. In particular, we do not embed the finite horizon MDP into an infinite horizon one. The solution procedure that one obtains using the approach in [13] is at best only approximate, a restriction not applicable to our work. In the limit as the number of updates goes to infinity, our algorithms converge to the optimal T-stage finite horizon policy. Our algorithms update all components of the policy vector at every update epoch, resulting in the near-equal convergence behaviour of $r-$th stage policy components and costs-to-go, for each $r \in \{0, 1, ..., T-1\}$.

Further, the method of [13] is a trajectory-based scheme, in that repeated simulations of entire $T-$length trajectories are performed, whereas our algorithm uses single transitions. In the former method, not all (state,action) pairs are sufficiently explored and hence a separate exploration function is needed. Apart from a look-up table proportional in the number of actions per-state, the algorithm of [13] also requires counters for the number of times a (state,action) pair has been seen by the algorithm.

Section II describes the framework of a finite-horizon MDP, provides a brief background of the techniques used, and proposes the algorithm. In section III, we illustrate numerical experiments in the framework of flow control in communication networks and capacity switching in semiconductor fabrication wherein we compute the optimal costs and policies using the proposed algorithm and compare performance with Dynamic Programming using certain performance metrics. We dwell on some future directions in section IV.

II. Framework and Algorithms

Consider an MDP $\{X_r, r = 0, 1, ..., T\}$ with decision horizon $T < \infty$. Suppose $\{Z_r, r = 0, 1, ..., T-1\}$ be the associated control valued process. Decisions are made at instants $r = 0, 1, ..., T-1$, and the process terminates at instant T. Let state space at epoch r be $S_r, r = 0, 1, ..., T$ and let the control space at epoch r be $C_r, r = 0, 1, ..., T-1$. Note that S_T is the set of terminating states of this process. Let $U_r(i_r) \subset C_r, r = 0, 1, ..., T-1$, be the set of all feasible controls in state i_r, in period r. Let $p_r(i, a, j), i \in S_r, a \in U_r(i), j \in S_{r+1}, r = 0, 1, ..., T-1$ denote the transition probabilities associated with this MDP. The transition dynamics of this MDP is governed according to

$$P(X_{r+1} = i_{r+1} | X_k = i_k, Z_k = a_k, 0 \leq k \leq r) =$$

$$p_r(i_r, a_r, i_{r+1})$$

$r = 0, 1, ..., T-1$, for all $i_0, i_1, ..., i_T, a_0, a_1, ..., a_{T-1}$, in appropriate sets. We define an admissible policy π as a set of T functions $\pi = \{\mu_0, \mu_1, ..., \mu_{T-1}\}$ with $\mu_r : S_r \longmapsto C_r$ such that $\mu_r(i) \in U_r(i)$, $\forall i \in S_r, r = 0, 1, ..., T-1$. Thus at (given) instant r with the system in say state i, the controller under policy π selects the action $\mu_r(i)$. Let $g_r(i, a, j)$ denote the single stage cost at instant r when state is $i \in S_r$, the action chosen is $a \in U_r(i)$ and the subsequent next state is $j \in S_{r+1}$, respectively, for $r = 0, 1, ..., T-1$. Also, let $g_T(k)$ denote the terminal cost at instant T when the terminating state is $k \in S_T$. The aim here is to find an admissible policy $\pi = \{\mu_0, \mu_1, ..., \mu_{T-1}\}$ that minimizes for all $i \in S_0$,

$$V_0^i(\pi) =$$

$$E\left\{ g_T(X_T) + \sum_{r=0}^{T-1} g_r(X_r, \mu_r(X_r), X_{r+1}) | X_0 = i \right\}. \quad (1)$$

The expectation above is over the joint distribution of $X_1, X_2, ..., X_T$. The dynamic programming algorithm for this problem is now given as follows (see [14]): for all $i \in S_T$,

$$V_T^i(\pi) = g_T(i) \quad (2)$$

and for all $i \in S_r$, $r = 0, 1, ..., T-1$,

$$V_r^i(\pi) =$$

$$\min_{a \in U_r(i)} \left\{ \sum_{j \in S_{r+1}} p_r(i, a, j)(g_r(i, a, j) + V_{r+1}^j(\pi)) \right\}, \quad (3)$$

respectively.

A. Brief Overview and Motivation

Note that using dynamic programming, the original problem of minimizing, over all admissible policies π, the T-stage cost-to-go $V_0^i(\pi)$, $\forall i \in S_0$, as given in (1), is broken down into T coupled minimization problems given by (2)-(3) with each such problem defined over the corresponding feasible set of actions for each state. Here,'coupled' means that the $r-$th problem depends on the solution of the $(r+1)-$th problem, for $0 \leq r < T$. In this paper, we are interested in scenarios where the $p_r(i,a,j)$ are not known, however, where transitions can be simulated. Thus, given that the state of the system at epoch r ($r \in \{0,1,..,T-1\}$) is i and action a is picked (possibly randomly), we assume that the next state j can be obtained through simulation.

We combine the theories of two-timescale stochastic approximation and SPSA to obtain actor-critic algorithms that solve all the T coupled minimization problems (2)-(3), under the (above) lack of model information constraint. For the case of infinite horizon problems, [3], [4] and [5] all use two-timescale stochastic approximation algorithms of which [5] adopts two-simulation SPSA to estimate the policy gradient.

Before we proceed further, we first motivate the use of SPSA based gradient estimates and two timescales in our algorithm. Suppose we are interested in finding the minimum of a function $F(\theta)$ when F is not analytically available, however, noisy observations $f(\theta,\xi_n)$, $n \geq 0$ of $F(\theta)$ are available with ξ_n, $n \geq 0$ being i.i.d. random variables satisfying $F(\theta) = E[f(\theta,\xi_n)]$. The expectation above is taken w.r.t. the common distribution of ξ_n, $n \geq 0$. Let $\theta = (\theta_1,...,\theta_q)^T$, $\Delta_i(n)$, $i = 1,...,q$, $n \geq 0$ be generated according to the method in section II-A.1 below and let $\Delta(n) = (\Delta_1(n),\ldots,\Delta_N(n))^T$, $n \geq 0$. Let $\theta(n)$ denote the nth update of parameter θ. For a given (small) scalar $\delta > 0$, form the parameter vector $\theta(n) + \delta\Delta(n)$. Then the one-measurement SPSA gradient estimate $\tilde{\nabla}_i F(\theta(n))$ of $\nabla_i F(\theta(n))$, $i = 1,\ldots,q$ has the form (cf. algorithm SPSA2-1R of [8]:

$$\tilde{\nabla}_i F(\theta(n)) = \frac{f(\theta(n)+\Delta(n),\xi_n)}{\delta\Delta_i(n)}, \qquad (4)$$

Note that only one measurement (corresponding to $\theta(n) + \delta\Delta(n)$) is required here.

Also observe that unlike SPSA estimates, Kiefer-Wolfowitz gradient estimates require $2q$ (resp. $(q+1)$) measurements when symmetric (resp. one-sided) differences are used. We now describe the construction of the deterministic perturbations $\Delta(n)$, $n \geq 1$, as proposed in [8].

1) Construction for Deterministic Perturbations: Let H be a normalized Hadamard matrix (a Hadamard matrix is said to be normalized if all the elements of its first row and column are 1s of order P with $P \geq q+1$). Let $h(1),...,h(q)$ be any q columns other than the first column of H, thus forming a new $(P \times q)-$ dimensional matrix $\hat{H}$. Let $\hat{H}(p)$, $p = 1,...,P$ denote the P rows of $\hat{H}$. Now set $\Delta(n) = \hat{H}(n \bmod P + 1)$, $\forall n \geq 0$. The perturbations are thus generated by cycling through the rows of the matrix $\hat{H}$. Here P is chosen as $P = 2^{\lceil \log_2 (q+1) \rceil}$. It is shown in [8] that under the above choice of P, the bias in gradient estimates asymptotically vanishes. Finally, matrices $H \equiv H_{P\times P}$ of dimension $P \times P$, for $P = 2^k$, are systematically constructed as follows:

$$H_{2\times 2} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

$$H_{2^k \times 2^k} = \begin{pmatrix} H_{2^{k-1}\times 2^{k-1}} & H_{2^{k-1}\times 2^{k-1}} \\ H_{2^{k-1}\times 2^{k-1}} & -H_{2^{k-1}\times 2^{k-1}} \end{pmatrix},$$

for $k > 1$.

2) Two-timescale stepsizes: Let $\{b(n)\}$ and $\{c(n)\}$ be two step-size schedules that satisfy

$$\sum_n b(n) = \sum_n c(n) = \infty, \quad \sum_n b(n)^2, \sum_n c(n)^2 < \infty,$$

and

$$c(n) = o(b(n)),$$

respectively. Thus $\{c(n)\}$ goes to zero faster than $\{b(n)\}$ does and corresponds to the slower timescale (since beyond some integer N_0 (i.e., for $n \geq N_0$), the sizes of increments in recursions that use $\{c(n)\}$ are uniformly the smallest, and hence result in slow albeit graceful convergence). Likewise, $\{b(n)\}$ is the faster scale. Informally, a recursion having $b(n)$ as the stepsize views a recursion that has stepsize $c(n)$ as static whilst the latter recursion views the former as having converged.

B. Proposed Algorithm (RPAFA)

The acronym RPAFA stands for 'Randomized Policy Algorithm over Finite Action sets'. In the following, for notational simplicity and ease of exposition, we assume that the state and action spaces are fixed and do not vary with stage. Thus S and C respectively denote the state and control spaces. Further, $U(i)$ denotes the set of feasible actions in state $i \in S$ and is also stage invariant. The set $U(i)$ of feasible actions in state i is assumed finite. Further, we assume that each set $U(i)$ has exactly $(q+1)$ elements $u(i,0)$, ..., $u(i,q)$ that however may depend on state i. For theoretical reasons, we will need the following assurance on per-stage costs.

Assumption (A) The cost functions $g_r(i,a,j), i,j \in S$, $a \in U(i)$, are bounded for all $r = 0,1,...,T-1$. Further, $g_T(i)$ is bounded for all $i \in S$.

We now introduce a Randomized Policy (RP). Let $\pi_r(i,a)$ be the probability of selecting action a in state i at instant r and let $\hat{\pi}_r(i)$ be the vector $(\pi_r(i,a), i \in S, a \in U(i)\backslash\{u(i,0)\})^T$, $r = 0,1,...,T-1$. Thus given $\hat{\pi}_r(i)$ as above, the probability $\pi_r(i,u(i,0))$ of selecting action $u(i,0)$ in state i at instant r is automatically specified as $\pi_r(i,u(i,0)) = 1 - \sum_{j=1}^q \pi_r(i,u(i,j))$. Hence, we identify a RP with $\hat{\pi} = (\hat{\pi}_r(i), i \in S, 0 \leq r \leq T-1)^T$. Let $\bar{S} = \{(y_1,\ldots,y_q) | y_j \geq 0, \forall j = 1,\ldots,q, \sum_{j=1}^q y_j \leq 1\}$ denote the simplex in which $\hat{\pi}_r(i)$, $i \in S$, $r = 0,1,...,T-1$ take values. Suppose $P : \mathcal{R}^q \mapsto \bar{S}$ denotes the projection map that projects $\hat{\pi}_r(i)$ to the simplex $\bar{S}$ after each update of the algorithm below.

Algorithm RPAFA

- *Step 0 (Initialize): Fix $\hat{\pi}_{0,r}(i,a)$, $\forall i \in S$, $a \in U(i)$, $0 \leq r \leq T-1$ as the initial RP iterate. Fix integers L and (large) M arbitrarily. Fix a (small) constant $\delta > 0$. Choose step-sizes $b(n)$ and $c(n)$ as in Section II-A.2. Generate $\hat{H}$ as in Section II-A.1. Set $V_{k,r}(i) = 0$, $\forall 0 \leq r \leq T-1$, and $V_{k,T}(i) = g_T(i)$, $0 \leq k \leq L-1$, $i \in S$ as initial estimates of cost-to-go. Set 'actor' index $n := 0$.*
- *Step 1: For all $i \in S$, $0 \leq r \leq T-1$, do:*
  - *Set $\Delta_{n,r}(i) := \hat{H}(n \bmod P + 1)$,*
  - *Set $\bar{\pi}_{n,r}(i) := P(\hat{\pi}_{n,r}(i) + \delta\Delta_{n,r}(i))$.*

- *Step 2 (Critic): For 'critic' index* $m = 0, 1, ..., L-1$, $r = T-1, T-2, ..., 0$, *and* $i \in S$, *do*
  - *Simulate action* $\phi_{nL+m,r}(i)$ *according to distribution* $\bar{\pi}_{n,r}(i)$.
  - *Simulate next state* $\eta_{nL+m,r}(i)$ *according to distribution* $p_r(i, \phi_{nL+m,r}(i), \cdot)$.
  - $V_{nL+m+1,r}(i) := (1 - b(n))V_{nL+m,r}(i)$
$$+b(n)g_r(i, \phi_{nL+m,r}(i), \eta_{nL+m,r}(i))$$
$$+b(n)V_{nL+m,r+1}(\eta_{nL+m,r}(i)).$$
- *Step 3 (Actor): For* $i \in S$, $0 \leq r \leq T-1$, *do*

$$\hat{\pi}_{n+1,r}(i) := P(\hat{\pi}_{n,r}(i) - c(n)\frac{V_{nL,r}(i)}{\delta}\Delta_{n,r}^{-1}(i))$$

  Set $n := n + 1$.
  If $n = M$, *go to Step 4;*
  else go to Step 1.
- *Step 4 (termination): Terminate algorithm and output* $\bar{\pi}_M$ *as the final policy.*

III. Simulation Results

A. Flow control in communication networks

We consider a continuous-time queuing model of flow control. The numerical setting here is somewhat similar to that in [5]. In [5], this problem is modelled in the infinite horizon discounted cost MDP framework, while we study it in the finite horizon MDP framework here. Flow and congestion control problems are suited to a finite horizon framework since a user typically holds the network for only a finite time duration. In many applications in communication networks, not only is the time needed to control congestion of concern, these applications must also be supported with sufficient bandwidth throughout this duration.

Assume that a single bottleneck node has a finite buffer of size B. Packets are fed into the node by both an uncontrolled Poisson arrival stream with rate $\lambda_u = 0.2$, and a controlled Poisson process with rate $\lambda_c(t)$ at instant $t > 0$. Service times at the node are i.i.d., exponentially distributed with rate 2.0. We assume that the queue length process $\{X_t, t > 0\}$ at the node is observed every $\tilde{T}$ instants, for some $T > 0$, upto the instant $\tilde{T}T$. Here T stands for the terminating stage of the finite horizon process. Suppose X_r denotes the queue length observed at instant $r\tilde{T}$, $0 \leq r \leq T$. This information is fed back to the controlled source which then starts sending packets at $\lambda_c(X_r)$ in the interval $[r\tilde{T}, (r+1)\tilde{T})$, assuming there are no feedback delays. We use $B = 50$ and $T = 10$, and designate the 'target states' $\hat{T}_r$ as evenly spaced states within the queue i.e., $\{\hat{T}_1 = 40, \hat{T}_2 = 37, \hat{T}_3 = 34, ..., \hat{T}_9 = 16, \hat{T}_{10} = 0\}$. The one-step transition cost under a given policy π is computed as $g_r(i_r, \phi_r(i_r), i_{r+1}) = \left|i_{r+1} - \hat{T}_{r+1}\right|$ where $\phi_r(i_r)$ is a random variable with law $\pi_r(i_r)$. Also, $g_T(i) = 0, \forall i \in S$. Such a cost function penalizes states away from the target states $\hat{T}_{r+1}$, apart from satisfying Assumption (A). The goal thus is to maximize throughput in the early stages (r small), while as r increases, the goal steadily shifts towards minimizing the queue length and hence the delay as one approaches the termination stage T.

For the finite action setting, we discretize the interval $[0.05, 4.5]$ so as to obtain five equally spaced actions in each state. For purposes of comparison, we also implemented the DP algorithm (1)-(2) for the finite action setting. Application of DP presupposes availability of the transition probability matrix $P_{\tilde{T}}$, and in order to compute these we use the approximation method of [15, §6.8]. Since each state has the same $q+1$ admissible controls, $q+1$ number of $P_{\tilde{T}}$ matrices of size $B \times B$ each are required. This storage required becomes prohibitive as either the state space increases, or the discretization is made finer. Also note that the amount of computation required for $P_{\tilde{T}}$ also depends upon the convergence criteria specified for the method in [15, §6.8]. Besides, such probabilities can only be computed for systems whose dynamics are well known, our setting being that of a well-studied $M/M/1/B$ queue.

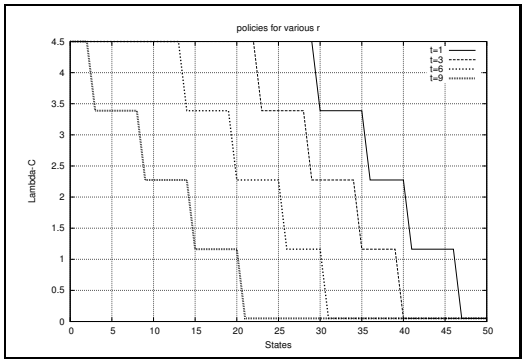

Fig. 1. Optimal Policy computed Using DP

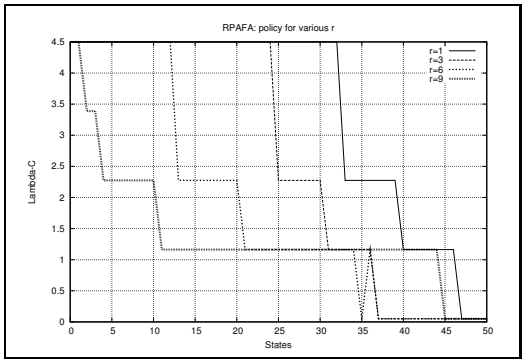

Fig. 2. Policy computed using RPAFA

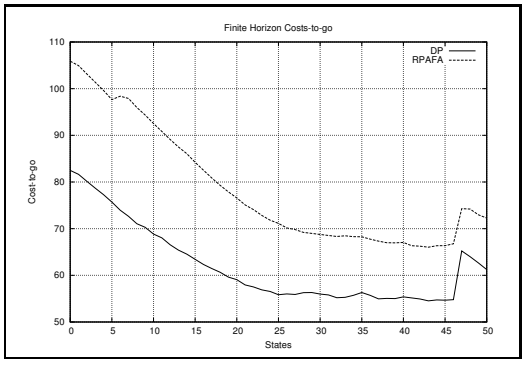

Fig. 3. Comparison of Costs-to-go

The policy obtained using finite-horizon DP with $P_{\tilde{T}}$ computed as above is shown in Figure 1. Note how the policy graphs shift to the left as the target state moves to the left for increasing r. Thus, the throughput of the

	r=2	r=6	r=10
Target $\hat{T}_r$	37	25	0
DP	27.1±6.3	22.4±5.1	5.7±3.6
RPAFA	19.2±6.5	19.4±4.8	8.9±5.3

TABLE I

OBSERVED $E(i_r)$ FOR THE PROPOSED ALGORITHM

	r=2	r=6	r=10
Target $\hat{T}_r$	37	25	0
DP	0.07±0.13	0.21±0.34	0.19±0.31
RPAFA	0.01±0.03	0.13±0.22	0.14±0.24

TABLE II

PROBABILITIES $p_r = P(i_r = \hat{T}_r \pm 1)$

system decreases from one stage to another as the target queue length is brought closer to zero.

The algorithm RPAFA is terminated at iteration n where $err_n \leq 0.01$. The convergence criterion is

$$err_n = \max_{i \in S, k \in \{1,2,\ldots,50\}} \|\pi_n(i) - \pi_{n-k}(i)\|_2,$$

where $\pi_n(i) = (\pi_{n,r}(i,a), 0 \leq r \leq T-1, a \in U(i))^T$. Further, $\|\cdot\|_2$ is the Euclidean norm in $\mathcal{R}^{T \times (q+1)}$. On a Pentium III computer using the C programming language, termination required upto 47×10^3 updates and 8×10^3 seconds. The policy obtained is shown in Figure 2, where for each state the source rate indicated is the rate that has the maximum probability of selection. Also shown in Figure 3 is the finite-horizon cost for each state in a system operating under the policies in Figure 1 and Figure 2, respectively. The plots shown in Figure 3 are obtained from 2×10^5 independent sample trajectories $\{i_0, i_1, ..., i_{10}\}$, each starting from state $i_0 = 0$ with a different initial seed. Tables I, II and III show performance comparisons of the proposed algorithm for various metrics with mean and standard deviation taken over the (above mentioned) 2×10^5 independent sample trajectories. Table I shows the mean queue length $E(i_r)$ at instants $r = 2, 4, 6, 8,$ and 10, respectively, with the corresponding standard deviation. With $\hat{T}_r$ defined as the 'target' state at instant r, Table II shows the probability of the system being in states $\hat{T}_r \pm 1$ at the above values of r. Table III shows the mean one-stage cost $E(g_{r-1}(i_{r-1}, \mu_{r-1}(i_{r-1}), i_r)) = E(|i_r - \hat{T}_r|)$ incurred by the system during transition from i_{r-1} to i_r under policy π. Note that the relatively bad performance of RPAFA is since it applies a randomized policy wherein the optimal action, though having a high probability of selection, may not be selected each time.

The approximate DP algorithm in this case converges much faster since transition probabilities using the method in [15] are easily computed in this setting. However, in most real life scenarios, computing these probabilities may not be as simple, and one may need to rely exclusively on simulation-based methods.

	r=2	r=6	r=10
Target $\hat{T}_r$	37	25	0
DP	10.5±5.5	5.2±3.6	5.8±3.6
RPAFA	18.0±6.3	6.7±3.9	8.8±5.3

TABLE III

MEAN ONE-STAGE COSTS $E(|i_r - \hat{T}_r|)$

B. Capacity Switching in Semiconductor Fabs

We first briefly describe the model of a semiconductor fabrication unit, considered in [16]. The factory process $\{X_r | X_0 = i\}, 1 \leq r \leq T, i \in S_0$ is such that each X_r is a vector of capacities at time epochs $r \in \{1, 2, ..., T\}$, the components $X_r^{(A,i),w}$ representing the number of type w machines allocated to performing operation i on product A. The duration of the planning horizon is T, casting this problem as a finite horizon Markov Decision Process. A type w machine indicates a machine capable of performing all operations which are letters in the 'word' w. Note that the product A requires the operation i for completion and that the word w contains the letter i, among others. The word w of a machine can also contain the action 0 - indicating idling. The control $\mu_r(X_r)$ taken at stages $0 \leq r \leq T-1$ would be to switch $u_r^{(A,i),w,(B,j)}$ machines of type w from performing operation i on product A to performing operation j on product B.

As in inventory control models, the randomness in the system is modeled by the demand D_r^A for product A at stages $0 \leq r \leq T-1$. The per-step transition cost consists of costs for excess inventory (K_A^e, for every unit of product A in inventory), backlog (K_A^b), cost of operation (K_w^o, one-stage operating cost for a machine of type w) and the cost of switching capacity from one type of operation to another (K_w^s, the cost of switching a machine of type w from one type of production to another). In all these costs, further complications can be admitted, e.g., the cost K_w^s could be indexed with the source product-operation pair (A, i) and the destination pair (B, j).

We consider here a simple model also experimented with in [16] where the infinite horizon discounted cost for a semiconductor fab model was computed using function approximation coupled with the policy iteration algorithm. In contrast, we do not use function approximation and adopt a finite horizon of 10. The cost structure, however, remains the same as [16]. In particular, we consider a fab producing two products (A and B), both requiring two operations, 'litho' and 'etch' (call these l and e). We have a fab with 2 litho and 2 etch machines that can each perform the corresponding operation on either A or B. We denote the operations $\{A, l\}$ (i.e., 'litho on A') as 1, $\{A, e\}$ as 2, $\{B, l\}$ as 3 and $\{B, e\}$ as 4, implying that the word w of a litho machine is 013 and that of an etch machine is 024. The product-operation pairs that admit non-zero capacities are therefore: $(A, 1)$, $(A, 2)$, $(B, 3)$, and $(B, 4)$.

The fab's throughput $P_r = (P_r^A, P_r^B)$ is constrained by the following relation: $P_r^A = min(X_r^{(A,1),013}, X_r^{(A,2),024})$ and $P_r^B = 0.5 \cdot min(X_r^{(B,3),013}, X_r^{(B,4),024})$, $0 \leq r \leq T-1$ implying the slower production of B. We assume that no machine is idling and therefore the capacity allocated to product B, given capacity allocated to A, is simply the remainder from 2 for both litho and etch. Therefore, these are $X_r^{(B,3),013} = 2 - X_r^{(A,1),013}$, $X_r^{(B,4),024} = 2 - X_r^{(B,4),024}$. We also constrain the inventories: $I_r^A \in \{-1, 0, 1\}$, $I_r^B \in \{-0.5, 0, 0.5\}$, $0 \leq r \leq T-1$. The state is thus completely described by the quadruplet: $X_r = (X_r^{(A,1),013}, X_r^{(A,2),024}, I_r^A, I_r^B), 0 \leq r \leq T$. One can choose a lexicographic order among the possible starting states X_0, which are $3^4 = 81$ in number.

Capacity switching at stage $0 \leq r \leq T-1$ is

	Iterations	Time in sec.	err_n	Max Error
RPAFA	1500	209	0.1	30.4

TABLE IV

PERFORMANCE OF RPAFA

specified by the policy component μ_r. Thus $\mu_r(X_r)$ is a vector such that $\mu_r(X_r) = (\mu_r(X_r, 1), \mu_r(X_r, 2))$, making $X_{r+1}^{(A,1),013} = X_r^{(A,1),013} + \mu_r(X_r, 1)$ (similarly for $X_{r+1}^{(A,2),024}$). Note that controls $\mu_r(X_r)$ belong to the feasible action set $U_r(X_r)$, and that in our setting $0 \leq |U_r(X_r)| \leq 9$, the state 'group' $(1, 1, x_A, y_B)$, $\forall x_A \in \{-1, 0, 1\}$, $\forall y_B \in \{-0.5, 0, 0.5\}$ having the maximum of 9 actions. We take the various one-step costs to be $K_A^e = 2$, $K_B^e = 1$, $K_A^b = 10$, $K_B^b = 5$, $K_{013}^o = 0.2$, $K_{024}^o = 0.1$, $K_{013}^s = 0.3$ and $K_{024}^s = 0.3$. The noise (demand) scheme is such that: $D_r^A = 1$ w.p. 0.4, $D_r^A = 2$ otherwise, and $D_r^B = 0.5$ w.p. 0.7, $D_r^B = 1$ otherwise, for all $0 \leq r \leq T-1$. We assume that the control $\mu_r(X_r)$ is applied to the state X_r at the beginning of the epoch r whereas the demand D_r is made at the end of it, i.e., $X_{r+1}^{(A,1),013} = X_r^{(A,1),013} + \mu_r(X_r, 1)$, $X_{r+1}^{(A,2),024} = X_r^{(A,2),024} + \mu_r(X_r, 2)$, $I_{r+1}^A = max(min(I_r^A + P_r^A - D_r^A, 1), -1)$, $I_{r+1}^A = max(min(I_r^B + P_r^B - D_r^B, 0.5), -0.5))$. We also take the terminal cost $K_T(X_T)$ to be 0. Thus, we have the per-stage cost:

$$K_r(X_r, \mu_r(X_r), X_{r+1}) =$$
$$2K_{013}^o + 2K_{024}^o + \mu_r^{A,1} \cdot K_{013}^s + \mu_r^{A,2} \cdot K_{024}^s$$
$$+max(I_{r+1}^A, 0) \cdot K_A^e + max(I_{r+1}^B, 0) \cdot K_B^e$$
$$+max(-I_{r+1}^A, 0) \cdot K_A^b + max(-I_{r+1}^B, 0) \cdot K_B^b$$

The resulting costs-to-go obtained using our algorithm is plotted in Figure 4. The states are sorted in decreasing order of the exact costs-to-go (computed using DP). The corresponding cost computed using the converged policy of RPAFA is seen to be reliably close - albeit higher than DP. The computing performance is outlined in Table IV. In the experiments we chose L and δ as 200 and 0.1, respectively.

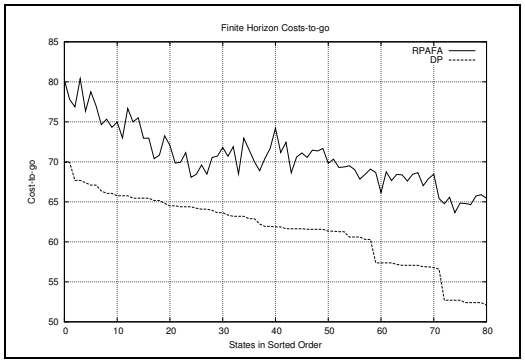

Fig. 4. Comparison of Costs-to-go

IV. FUTURE DIRECTIONS

The policy iteration algorithm used one-simulation SPSA gradient estimates with perturbation sequences derived from normalized Hadamard matrices. However, in general, two-simulation SPSA algorithms are seen to perform better and converge faster as compared to one-simulation SPSA, which could be derived along similar lines (see [8]). In order to further improve performance, efficient simulation-based higher order SPSA algorithms that also estimate the Hessian in addition to the gradient along the lines of [17] could also be explored.

Acknowledgments

This work was supported in part by Grant no. SR/S3/EE/43/2002-SERC-Engg from the Department of Science and Technology, Government of India.

REFERENCES

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

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

[3] V. Konda and V. Borkar, "Actor–Critic Type Learning Algorithms for Markov Decision Processes," *SIAM Journal on Control and Optimization*, vol. 38, no. 1, pp. 94–123, 1999.

[4] V. Konda and J. Tsitsiklis, "Actor–Critic Algorithms," *SIAM Journal on Control and Optimization*, vol. 42, no. 4, pp. 1143–1166, 2003.

[5] S. Bhatnagar and S. Kumar, "A Simultaneous Perturbation Stochastic Approximation–Based Actor–Critic Algorithm for Markov Decision Processes," *IEEE Transactions on Automatic Control*, vol. 49, no. 4, pp. 592–598, 2004.

[6] J. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, vol. 37, no. 1, pp. 332–341, 1992.

[7] ——, "A One–Measurement Form of Simultaneous Perturbation Stochastic Approximation," *Automatica*, vol. 33, no. 1, pp. 109–112, 1997.

[8] S. Bhatnagar, M. Fu, S. Marcus, and I.-J. Wang, "Two–timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences," *ACM Transactions on Modeling and Computer Simulation*, vol. 13, no. 4, pp. 180–209, 2003.

[9] H. Chang, P. Fard, S. Marcus, and M. Shayman, "Multitime Scale Markov Decision Processes," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 976–987, 2003.

[10] Y. Serin, "A nonlinear programming model for partially observed Markov decision processes: finite horizon case," *European Journal of Operational Research*, vol. 86, pp. 549–564, 1995.

[11] A. Lin, J. Bean, and C. White, III, "A hybrid genetic/optimization algorithm for finite horizon partially observed Markov decision processes," *INFORMS Journal On Computing*, vol. 16, no. 1, pp. 27–38, 2004.

[12] C. Zhang and J. Baras, "A hierarchical structure for finite horizon dynamic programming problems," *Technical Report TR2000-53, Institute for Systems Research, University of Maryland, USA*, 2000.

[13] F. Garcia and S. Ndiaye, "A learning rate analysis of reinforcement learning algorithms in finite-horizon," in *Proceedings of the Fifteenth International Conference on Machine Learning, Madison, USA*, 1998.

[14] D. Bertsekas, *Dynamic Programming and Optimal Control, Volume I*. Belmont, MA: Athena Scientific, 1995.

[15] S. Ross, *Introduction to Probability Models, 7/e*. San Diego, CA: Academic Press, 2000.

[16] Y. He, S. Bhatnagar, M. Fu, S. Marcus, and P. Fard, "Approximate policy iteration for semiconductor fab-level decision making - a case study," *Technical Report, Institute for Systems Research, University of Maryland*, 2000.

[17] S. Bhatnagar, "Adaptive multivariate three–timescale stochastic approximation algorithms for simulation based optimization," *ACM Transactions on Modeling and Computer Simulation*, vol. 15, no. 1, pp. 74–107, 2005.