Proceedings of the 2007 American Control Conference
Marriott Marquis Hotel at Times Square
New York City, USA, July 11-13, 2007

Parametrized Actor-Critic Algorithms for Finite-Horizon MDPs

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

***Abstract*— Due to their non-stationarity, finite-horizon Markov decision processes (FH-MDPs) have one probability transition matrix per stage. Thus the curse of dimensionality affects FH-MDPs more severely than infinite-horizon MDPs. We propose two parametrized 'actor-critic' algorithms to compute optimal policies for FH-MDPs. Both algorithms use the two-timescale stochastic approximation technique, thus simultaneously performing gradient search in the parametrized policy space (the 'actor') on a slower timescale and learning the policy gradient (the 'critic') via a faster recursion. This is in contrast to methods where critic recursions learn the cost-to-go proper. We show w.p 1 convergence to a set with the necessary condition for constrained optima. The proposed parameterization is for FH-MDPs with compact action sets, although certain exceptions can be handled. Further, a third algorithm for stochastic control of stopping time processes is presented. We explain why current policy evaluation methods do not work as critic to the proposed actor recursion. Simulation results from flow-control in communication networks attest to the performance advantages of all three algorithms.**

Keywords

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

I. Introduction

Policy evaluation using parameterized cost-to-go estimates in Markov Decision Processes (MDPs) are relevant due to their wide applications in communication, finance and control where state-spaces of the systems are typically large. In particular, policy evaluation algorithms that approximate the cost-to-go from each state using linear parameterization have been proposed for the infinite-horizon criteria of average cost [1] and discounted cost [2] and further for stopping time problems [3]. A crucial enhancement to the basic method is pursued in [4]. However, with regard to finite-horizon total cost MDPs (F-H MDPs), there has not been any treatment of a similar nature. This may be due to the convenience that full $T-$ length trajectories can be simulated in F-H MDPs, without having to resort to use of the key Poisson equation. Such an advantage is not available to infinite-horizon MDPs, however, where parametrized cost-to-go estimates are further employed to compute the optimal policy using a policy gradient method (cf. [5]).

For F-H MDPs, a simulation-based solution that stored the cost-to-go estimates in a look-up table was proposed in [6]. A key feature of the scheme of [6] is that online sampling of the states is not necessary, i.e. simulating an entire trajectory is not required, repeated simulation of a single transition from each state being sufficient. However, at issue here is the size of such a look-up table, since when compared to infinite-horizon MDPs, the memory requirements in F-H MDPs assume severity due to the absence of stationarity. A look-up table of size $|S| \times T$ is needed, where $T \in \mathcal{Z}^+$ is the length of the finite horizon and assuming that all states are feasible in all stages. It is this memory requirement that the proposed algorithms address.

Algorithms for finding optimal policy using Simultaneous Perturbation Stochastic Approximation (SPSA) to estimate the policy gradient while minimizing the infinite-horizon average-cost, were proposed in [7]. Many F-H MDPs require only a finite-action setting, but since the closed *convex-hull* of such an action set is compact, the algorithm DPAFA of [7] is applicable.

A. Outline of article and Notation

The linear parameterization of policy iterate θ_k is considered in Sections II and III and certain key differences w.r.t. relevant algorithms are also identified. Presented in Section II is a result that shows convergence w.p. 1 to a set satisfying necessary conditions of constrained optima. Further, Section III reports another algorithm for FH-MDPs with higher storage requirements, but lesser computation. Section IV considers extension to the control of stopping-time MDPs. Simulation results are presented in Section V and we conclude and identify some future directions in Section VI

For an FH-MDP operating under the $k-$th policy iterate θ_k, the finite-horizon cost-to-go from the $l-$th stage onward, $V_l(\theta_k, i)$, is given by:

$$E\{\sum_{m=l}^{T-1} K_m(X_m, \theta_{k,m}(X_m)) + K_T(X_T)|X_l = i\}.$$

This we abbreviate as $V_{k,l}(i)$. Here $K_m(\cdot,\cdot)$ denotes the single stage cost at instant m. Without loss of generality we assume that the terminal cost $K_T(X_T) = 0, \forall X_T \in S$. In the compact action setting, the action taken according to policy iterate θ_k when state i is observed in stage l is such that $\theta_{k,l}(i) \in U_{l,i} \stackrel{\triangle}{=} \Pi_{r=1}^{N_{l,i}}[a_{l,i}(r), b_{l,i}(r)]$ where $a_{l,i}(j) < b_{l,i}(j) \in \mathcal{R}, \forall i \in S$. Thus the constraint sets $U_{l,i}$ are chosen to be compact rectangles in $\mathcal{R}^{l,i}$. We operate under a constraint on the action set:

Assumption 1: $U_{l,i}$ are s.t. $N_{l,i} = N_l$, $a_{l,i}(r) = a_l(r)$, and $b_{l,i}(r) = b_l(r)$, $\forall i \in S$ and $1 \leq r \leq N_l$.

While this does make the set $U_{l,i}$ same over all $i \in S$, we claim in Section II that an analogous restriction is implicit in other parametrization methods proposed in the literature. Further, $\theta_{k,l} \in U_l \equiv \Pi_{i=1}^{|S|} U_{l,i}$ and $\theta_k \in U \stackrel{\triangle}{=} \Pi_{l=0}^{T-1} U_l$ where θ_k is the vector $(\theta_{k,l})_{l=0}^{T-1}$ and each $\theta_{k,l}$ is the vector $(\theta_{k,l}(i))_{i \in S}$, respectively. A projection operator $P_{l,i}(\theta_{k,l}(i))$ is used to truncate action $\theta_{k,l}(i)$ into the above feasible set $U_{l,i}$, with analogously defined operators P_l and P to project into U_l and U, respectively. The proposed SPSA actor recursion is subject to:

Assumption 2: For $i \in S$, $a \in U_{l,i}$, $0 \leq l \leq T-1$, and $j \in S$, costs $K_l(i,a)$ and transition probabilities $p_l(i,a,j)$ are continuously differentiable w.r.t. a.
The policy iterate θ_k is of size $|S| \times T$ (taking $N_{l,i} = 1$, $\forall l, i$) which is also the size of the look-up table $V_k = (V_{k,l}, 1 \leq l \leq T-1)^T$ where $V_{l,k} = (V_{l,k}(i), 1 \leq i \leq |S|)^T$. Since $N_{l,i}$ indicates the dimension of the action at stage l in state i, note that $N_{l,i} > 1$ would imply that size of policy θ_k is larger than that of the look-up table V_k.

In the spirit of [2] and related work, the proposed algorithm uses feature vectors $\phi_l(i), \forall i \in S$ where $\phi_l(i) \in \mathcal{R}^K, K \ll |S|$. Define the $|S| \times K$ matrix Φ_l as $(\phi_l(i), \forall i \in S)^T$ and the feature vector $\phi_l(i)$ for each stage-state pair l, i as $(\phi_{l,k}(i),\ 1 \leq k \leq K)^T$. These features ϕ_l - which can be constructed based on metrics to grade the states - become available every time the state is encountered and thus do not need to be stored. Feature vectors of states change with the stage $0 \leq l \leq T-1$, demonstrating an added complication in FH-MDPs when compared to infinite-horizon MDPs. Also, we approximate the gradient-estimate of the cost-to-go $\tilde{\bigtriangledown} V_{k,l} = (\tilde{\bigtriangledown}_i V_{k,l}(i), 1 \leq i \leq |S|)^T$ using a linear architecture, where, $\tilde{\bigtriangledown}_m V_{k,l}(n)$ is an estimate of the partial-gradient of $V_{k,l}(n)$ w.r.t. $\theta_{k,l}(m)$. Since $V_{k,l}(i) = K_l(i, \theta_{k,l}(i)) + \sum_{j \in S_{l+1}} p_l(i, \theta_{k,l}(i), j) V_{k,l+1}(j)$, terms $\bigtriangledown_m V_{k,l}(n)$, for $m, n \neq i$, do not contribute to the update of $\theta_{k,l}(i)$. For a parsimonious representation, $\tilde{\bigtriangledown} V_{k,l}$ is projected using the L^2 norm onto the space spanned by columns of Φ_l. The orthogonal projection of a vector $x \in \mathcal{R}^s$ is $\Pi_{\Phi_l} x$ where $\Pi_{\Phi_l} = \Phi_l(\Phi_l^T \Phi_l)^{-1} \Phi_l^T$. Thus $\tilde{\bigtriangledown} V_{k,l}$ is approximated by $\Pi_{\Phi_l} \tilde{\bigtriangledown} V_{k,l}$. All further treatment makes two simplifying assumptions, $N_{l,i} = 1, \forall l, i$ and that any state can be visited by the process in any stage.

II. FH-MDP Algorithm 1

Certain key differences with the algorithms currently in literature are outlined before the algorithm is proposed, including disadvantages of the proposed scheme:

- The linear parametrization of the cost-to-go in [2], [5] uses the fact that the vector $V_{k,l} \in \mathcal{R}^{|S|}$ (in the setting of [2], however, the subscript l is irrelevant). Assumption 1, where $U_l \subset \mathcal{R}^{|S|}$, helps us in similarly parametrizing the actor, i.e., the policy iterate θ_k.
- In the proposed algorithm, the critic used is an estimate of the gradient of the cost-to-go at policy iterate θ_k. This gradient estimate in $\mathcal{R}^{|S|}$ is then projected using operator Π_{Φ_l}. As seen in Theorem 1, this is central to showing convergence w.p. 1.
- The critic has the same linear parametrization as the actor, i.e. both belong to the span of Φ_l, although an estimate of the policy-gradient has no intuitive interpretation. We do not need to link the two parameterizations as in the scheme of [5]. Therefore, the proposed Algorithm 1 is an actor-critic implementation with a parametrized actor.
- Due to the compact action setting, two terms from [8], *a)* the likelihood ratio term $L_u(i,\theta)$, and *b)* gradient of reward term $\bigtriangledown K(i, \theta_k)$ are not required. Avoiding *b)* helps when one-step transition costs $K_l(X_l, \theta_{k,l}(X_l), X_{l+1})$ have the destination state X_{l+1} as an argument. We optimize directly the expected total cost and not the individual cost samples using perturbation-analysis type schemes as [8] does. As a result, we do not require constraining regularity conditions. While [8] also estimates performance gradient w.r.t. parameter θ_k (although for the infinite-horizon average-cost case), we use an explicit stochastic gradient formulation (viz. SPSA).
- The 'Monte Carlo' nature of the critic in Algorithm 1 - requiring a $(T-l)-$length simulated trajectory until horizon T is reached - is due to this inability to use the Poisson equation. Regeneration intervals, i.e. finite length simulated trajectories until a certain i^* is hit are also used in [8].
- In model-free algorithms, experimentation with a computer model of the system grants us liberty to simulate a $(T-l)-$length trajectory from any (stage, state) pair (l, i) even though it is the total cost-to-go $V_{0,i}$ that is to be minimized. We use this to good effect in the proposed algorithm. Note that this assumption is implicit in [6] and [9].
- No Dynamic Programming (DP) results are used in Algorithm 1. The Poisson equation: $V_{k,l}(X_l) = E\{K_l(X_l, \theta_{k,l}(X_l)) + V_{k,l+1}(X_{l+1})\}$, is avoided. Use of $\Pi_{\Phi_{l+1}} V_{k,l+1}$ in place of $V_{k,l+1}$ above renders the equation invalid, although this handicap is mitigated in Algorithm 2 of Section III later.
- Proposed algorithms only handle $U_{l,i} \equiv U_l$, $\forall i \in S$. An analogous restriction holds in, e.g., [5] since U_i are s.t. $|U_i| = u$, $\forall i \in S$. For an $i \in S$ where $N_{l,i} < N_l$, the proposed algorithm can yet be modified by adding dummy intervals, and scaling and translating $U_{l,i}$ that violate $U_{l,i} \equiv U_l$.

We next motivate the proposed algorithm. Similar to the Dynamic Programming algorithm, we update the current policy by moving backwards in horizon with index l from $T-1$ to 0 (although, crucially, we do not use the dynamic programming *principle*). We use the compact action set assumption, i.e., $\theta_{k,l}(i) \in U_{l,i}$, and $\theta_{k,l}(i) = P_{l,i}(\langle \phi_l(i), \bar{\theta}_{k,l} \rangle)$ for the parametrized policy iterate $\bar{\theta}_{k,l} \in \mathcal{R}^K$. However, the action $\theta_{k,l}(i)$ is not used explicitly. Instead, the two-sided SPSA method perturbs policy iterate $\theta_{k,l}$ with $\delta_{k,l}\Delta_{k,l}$ to produce policy $\theta^+_{k,l}(i) = P_{l,i}(\theta_{k,l}(i) + \delta_k \Delta_{k,l}(i))$ and $\theta^-_{k,l}(i) = P_{l,i}(\theta_{k,l}(i) - \delta_k \Delta_{k,l}(i))$ $\forall i \in S$ and measures system performance at these two policies. While δ_k is a perturbation parameter that diminishes to zero in order that the bias w.r.t. the true policy gradient at $\theta_{k,l}$ vanishes as $k \to \infty$ (cf. [10]), $\Delta_{k,l}$ is such that $\Delta_{k,l}(i) = \pm 1$ with probability 0.5 each. Policy gradient methods using SPSA estimates applied to solutions of MDPs are to be found in [6], [9] and [7].

The evaluation of $\Pi_{\Phi_l} \tilde{\bigtriangledown} V_{k,l}$ is undertaken in a manner reminiscent of the LSTD(0) algorithm in [4]. The projection of $V_{k,l}$ to the span of Φ_l is an optimization problem which LSTD(0) does not solve incrementally, unlike TD(λ) in [2]. Note that $\Pi_{\Phi_l} = \Phi_l(\Phi_l^T \Phi_l)^{-1} \Phi_l^T$ and hence we estimate $\Phi_l^T \tilde{\bigtriangledown} V_{k,l}$ first by making $L \gg 0$ starts from states i chosen uniformly out of S and simulating trajectory $\{X^+_l, X^+_{l+1}, ..., X^+_T | X^+_l = i\}$ using action $\theta^+_{k,l}(i)$ in stage l and state i and policies $\theta_{k+1,l+m}$, $m \geq 1$ for stages $l+m$ upto $T-1$. We call the accumulated cost in such a trajectory as K^+ and observe that since the starting state i is chosen uniformly from S, $|S|\phi_l(i) \tilde{\bigtriangledown}_i V_{k,l}(i)$ is a vector whose mean is $\Phi_l^T \tilde{\bigtriangledown} V_{k,l}$. The $K \times K$ matrix $(\Phi_l^T \Phi_l)^{-1}$ is assumed to be pre-stored but can also be estimated using an averaging of the $K \times K$ matrix iterates $\phi_l(i) \phi_l^T(i)$. Analogously, the simulation to compute K^- is performed. The averaging of the 'critic' estimates $|S|\phi_l(i)\tilde{\bigtriangledown}_i V_{k,l}(i)$ occurs on a faster timescale b_k than the scale a_k used to update the 'actor' $\theta_{k,l}$. This relationship results in the actor recursion viewing the critic as having equilibrated to $\Phi_l^T \tilde{\bigtriangledown} V_{k,l}$ at each update k. In the trajectory, the next state from X_t^+ is indicated by the simulation random variable $\eta(X_t^+, \theta_{k,t}^+(X_t^+))$. While neither $\theta_{k,l}$, $\theta_{k,l}^{\pm}$ nor $\theta_{k+1,l}$ may belong to the span of Φ_l, the availability of the projection operator P_l makes it possible to implement such policies during simulation. The number of policy updates M for which the algorithm below is run is typically decided using some convergence criteria, making θ_M the policy parameter output by the algorithm.

The requirements on step-sizes a_k and b_k are as follows:

$$b_k, a_k > 0, \forall k \geq 0 \quad , \quad \sum_k b_k = \sum_k a_k = \infty,$$

$$\sum_k {b_k}^2, \sum_k {a_k}^2 \quad < \quad \infty, \text{ and } a_k = o(b_k).$$

Similarly, the perturbation parameter δ_k is such that $\sum_k \frac{a_k}{\delta_k} = \infty$, $\sum_k \left(\frac{a_k}{\delta_k}\right)^2 < \infty$, and $\frac{a_k}{\delta_k} = o(b_k)$.

The proposed algorithm is described next:

```
• for $k = 0, 1, ..., M$ do: {
• for $l = T-1, ..., 0$ do: {
• Critic: for $m = 0, 1, ..., L-1$ do: {
    – choose start state $X_l^+ = X_l^- = i$ uniformly from $S$.
    – generate perturbation $\Delta_{k,l}(i)$
    – $K^+ := 0,\ t := l$.
    – while $t < T$ do: {
        if (t=l)
          $X_{l+1}^+ := \eta(i, P_{l,i}(\theta_{k,l}(i) + \delta_k \Delta_{k,l}(i)))$;
          $K^+ := K^+ + K_l(i, P_{l,i}(\theta_{k,l}(i) + \delta_k \Delta_{k,l}(i)))$;
        else
          $X_{t+1}^+ := \eta(X_t^+, P_{t,i}(\theta_{k,t}(X_t^+)))$;
          $K^+ := K^+ + K_t(X_t^+, P_{t,i}(\theta_{k,t}(X_t^+)))$;
        endif
        $t := t + 1$;
      }
    – $K^- := 0,\ t := l$.
    – while $t < T$ do: {
        if (t=l)
          $X_{l+1}^- := \eta(i, P_{l,i}(\theta_{k,l}(i) - \delta_k \Delta_{k,l}(i)))$;
          $K^- := K^- + K_l(i, P_{l,i}(\theta_{k,l}(i) - \delta_k \Delta_{k,l}(i)))$;
        else
          $X_{t+1}^- := \eta(X_t^-, P_{t,i}(\theta_{k,t}(X_t^-)))$;
          $K^- := K^- + K_t(X_t^-, P_{t,i}(\theta_{k,t}(X_t^-)))$;
        endif
        $t := t + 1$;
      }
    – $r_{kL+m+1,l} := r_{kL+m,l} + b_k \left(|S| \left(\frac{K^+ - K^-}{2\delta_k \Delta_{k,l}(i)}\right) \phi_l(i) - r_{kL+m,l}\right)$
  }
• Actor: $\bar{\theta}_{k+1,l} := (\Phi_l^T \Phi_l)^{-1} \Phi_l^T \cdot P_l\left(\Phi_l \bar{\theta}_{k,l} - a_k \Phi_l (\Phi_l^T \Phi_l)^{-1} r_{(k+1)L,l}\right)$   (1)
  }
}
```

We write (1) as: $\bar{\theta}_{n+1,l} := (\Phi_l^T \Phi_l)^{-1} \Phi_l^T \cdot$

$$P_l(\Phi_l \bar{\theta}_{n,l} - \frac{a_n}{2\delta_n}(V_{n,l}^+ - V_{n,l}^-) \circ \Delta_{n,l}^{-1}), \tag{2}$$

where $\circ$ stands for component-wise multiplication, in this case with the vector $\Delta_{k,l}^{-1}$ (this vector, incidentally, equals $\Delta_{k,l}$ due to our choice). The result below shows that, for a given stage l, the iterates in (1) converge to a candidate constrained optimum $\theta_l^* \in \text{span}(\Phi_l)$, discounting the possibility of spurious minima on the boundary of the feasible action set U_l. We adapt the algorithm in §5.1.5 of [11] which treats constrained function minimization using the well known Kiefer-Wolfowitz algorithm.

Theorem 1: $\theta_{k,l}$ converges w.p. 1 to a θ_l^* in a set of points with necessary condition for constrained optima i.e. $\{\theta_l^* : \Pi_{\Phi_l} \bigtriangledown_l V_l(\theta^*) = 0,\ \theta_l \in U_l;\ \forall l\}$, where $\theta^* = (\theta_0^*, \theta_1^*, ..., \theta_{T-1}^*)^T$.

Proof We reproduce the algorithm in §5.1.5 of [11] for clarity and explain the terms:

$$\begin{aligned} X_{n+1} \quad := \quad & X_n - a_n \pi(X_n) \left[Df(X_n, c_n) - \beta_n - \xi_n\right] \\ & - k a_n \Phi^T(X_n) \phi(X_n), \end{aligned} \tag{3}$$

where $X_n \in \mathcal{R}^r$ is the iterate and $Df(X_n, c_n)$ is a finite difference approximation to $\bigtriangledown_x f(X_n)$, the gradient at X_n of objective function $f : \mathcal{R}^r \mapsto \mathcal{R}$. The bias in such an estimate is represented by β_n whilst ξ_n is noise with certain conditions, Assumptions A5.1.1-A5.1.6 of [11] covering these. The constraint set is denoted as $\{x : \phi(x) = \underset{\sim}{0} \in \mathcal{R}^s, x \in \mathcal{R}^r\}$ where $\phi(x) \equiv (\phi_1(x), \phi_2(x), ..., \phi_s(x))^T$. Note that the ϕ above are not to be confused with feature vectors ϕ_l that we use. Further, $\Phi(X_n)$ is a matrix s.t. $\Phi^T(X_n) = (\bigtriangledown_x \phi_1(X_n), \bigtriangledown_x \phi_2(X_n), ..., \bigtriangledown_x \phi_s(X_n))$ (making $\Phi^T(X_n)$ an $r \times s$ matrix). The matrix $\pi(X_n)$ in (3) is obtained from projection $(I - \pi(X_n))$ to the span of the columns of $\Phi^T(X_n)$, whilst k is an arbitrary but fixed positive number.

We shall use the SPSA gradient estimate $\tilde{\bigtriangledown} V_{n,l} \stackrel{\triangle}{=} E\left(\frac{1}{2\delta_n}(V_{n,l}^+ - V_{n,l}^-) \circ \Delta_{n,l}^{-1} | \theta_n\right)$ in place of the $[\cdot]$ in (3). This is because replacing c_n with δ_n and using Lemma 1 of [10], the requirements in A5.1.11 of [11] are satisfied. We now rewrite (3) replacing iterates X_n with $\theta_{n,l}$ as follows: $\theta_{n+1,l} :=$

$$\theta_{n,l} - a_n \left[\pi(\theta_{n,l}) \tilde{\bigtriangledown}_l V_{n,l}\right] - k a_n \Phi^T(\theta_{n,l}) \phi(\theta_{n,l}). \tag{4}$$

The subscript l will distinguish our notation from the Φ and ϕ of (3) as we perform the following substitutions. The constraint set here is $\{\theta_l : (I - \Pi_{\Phi_l})\theta_l = 0\}$, indicating that $\theta_l \in \text{span}(\Phi_l)$. Similarly $\Phi^T(\theta_l)$ can be seen to be $(I - \Pi_{\Phi_l})^T$, $\forall \theta_l$. Observe that if $\theta_{n,l} \in \text{span}(\Phi_l)$ then $\phi(\theta_{n,l}) = 0$ canceling the last term in the RHS of (4). Further, $\pi(\theta_{n,l})$ is s.t. $(I - \pi(\theta_{n,l})) : \mathcal{R}^{|S|} \longrightarrow \mathcal{R}^{|S|}$ is a projection to the span of the columns of Φ^T, the symmetry of $I - \Pi_{\Phi_l}$ meaning that $\pi(\theta_{n,l}) = \Pi_{\Phi_l},\ \forall \theta_n$. Thus the recursion is now simplified to:

$$\bar{\theta}_{n+1,l} \quad = \quad \bar{\theta}_{n,l} - a_n \left[\Pi_{\Phi_l} \tilde{\bigtriangledown}_l V_{n,l}\right].$$

Observe that pre-multiplying the 'actor' with Φ_l in the proposed algorithm (1) yields the above equation. To conclude, the ODE evolution is also constrained to within U_l, via a transformed projection operator handled in, e.g., Theorem 1 of [7]. □

III. FH-MDP Algorithm 2: DP Principle

As before, the policy element $\theta_{l,k}$ is parametrized, i.e. it belongs to the span of Φ_l, $0 \leq l \leq T-1$ requiring $K \times T$ memory. We will require two $|S|-$sized look-up vectors, V^1 and V^2. For each index k of the algorithm, stages $l = T-1, T-2, ..., 0$, and all states $i \in S$ we generate the random variable $\Delta_{k,l}(i)$ from the set $\{+1,-1\}$ w.p. 0.5 each. Compute actions $P_l(\langle\phi_l(i), \theta_{k,l}\rangle \pm \delta_k \Delta_{k,l}(i))$ (call these $\theta^{\pm}_{k,l}(i)$), and simulate a transition using these actions to states η_l^+ and η_l^-, respectively, of the $(l+1)-$th stage. The array V^1 is interpreted as the vector of $(l+1)-$th stage costs-to-go, therefore $\tilde{\bigtriangledown}_i V_{k,l}(i) =$

$$\frac{1}{2\delta_k}\Delta^{-1}_{k,l}(i)\left(K_l(i, \theta^+_{k,l}(i), \eta_l^+) + V^1(\eta_l^+)\right.$$
$$\left. - K_l(i, \theta^-_{k,l}(i), \eta_l^-) - V^1(\eta_l^-)\right),$$

and we proceed further as in the critic recursion of the Algorithm 1, i.e. by scaling the vector $\phi_l(i)$ with the term $|S|\tilde{\bigtriangledown}_i V_{k,l}(i)$. In addition, we simulate η_l using $\theta_{k,l}(i) = P_l(\langle\phi_l(i), \theta_l\rangle)$ and update $V^2(i)$ using cost-to-go estimate $K_l(i, \theta_{k,l}(i), \eta_l) + V^1(\eta_l)$ in a recursion with stepsize b_k. After the actor update, we assign V^2 to V^1 and set V^2 to the zero vector $\underline{0}$. This algorithm is described next:

```
• for $k = 0, 1, ..., M$ do: {
• Initialize $V^1 := \underline{0}$
• for $l = T-1, ..., 0$ do: {
• Critic: for $m = 0, 1, ..., L-1$ do: {
• choose $i \in S$ uniformly, do: {
    – generate $\Delta_{k,l}(i)$
    – $X^+_{l+1} := \eta(i, P_{l,i}(\theta_{k,l}(i) + \delta_k \Delta_{k,l}(i)))$;
      $K^+ := K_l(i, P_{l,i}(\theta^+_{k,l}(i))) + V^1(X^+_{l+1})$;
    – $X^-_{l+1} := \eta(i, P_{l,i}(\theta_{k,l}(i) - \delta_k \Delta_{k,l}(i)))$;
      $K^- := K_l(i, P_{l,i}(\theta^-_{k,l}(i))) + V^1(X^-_{l+1})$;
    – $X_{l+1} := \eta(i, P_{l,i}(\theta_{k,l}(i)))$;
      $K := K_l(i, P_{l,i}(\theta_{k,l}(i))) + V^1(X_{l+1})$;
    – $V^2(i) := V^2(i) + b_k(K - V^2(i))$;
    – $r_{kL+m+1,l} := r_{kL+m,l} +$
      $b_k\left(|S|\left(\frac{K^+ - K^-}{2\delta_k \Delta_{k,l}(i)}\right)\phi_l(i) - r_{kL+m,l}\right)$
    }
    }
• Actor: $\bar{\theta}_{k+1,l} := (\Phi_l^T \Phi_l)^{-1}\Phi_l^T \cdot$
      $P_l\left(\Phi_l \bar{\theta}_{k,l} - a_k \Phi_l (\Phi_l^T \Phi_l)^{-1} r_{(k+1)L,l}\right)$
• Flip: $V^1 := V^2$ and $V^2 := \underline{0}$.
  }
  }
```

The advantage here is in the reduced simulation load: $O(T)$ instead of $O(T^2)$ in the previous algorithm. There is an increased memory requirement vis-a-vis the Algorithm 1, since we need additional $2|S|$ memory than previously. Thus, this method is suitable for FH-MDPs where $T \geq 2$ and actor parametrization may be permitted with only a tolerable loss in performance. Also note that the iterated backward flow $l = T-1, T-2, ..., 0$ of the proposed algorithm prevents parallelization of the algorithm, unlike analogous methods in [6].

We describe briefly another algorithm with reduced simulation load conceived of on the same lines. For each stage l moving backwards from $T-1$ to 0, $|S|$ times we pick a state $i \in S$ uniformly, simulating L transitions each using perturbed actions $\theta^+_{k,l}(i)$and $\theta^-_{k,l}(i)$, respectively. Note that we first choose i and then simulate $2L$ transitions from i, rather than pick states L times (and simulate 3 transitions from each, making for a total of $3L$) as was done in the previous algorithm. We update two *scalars* V^+ and V^- in the manner of $V^2(i)$ above, using K^+ and K^-, respectively, in place of K. After these L updates each of $V^{\pm}$, we update the parametrized policy-gradient r using estimate $\frac{V^+ - V^-}{2\delta_k \Delta_{k,l}(i)}$, followed by the assignment $V^2(i) := V^+ + V^-$. This last step uses Assumption 2 and the analysis of SPSA (cf. [10]), by which we have $E(V^+ + V^- | \theta_k, k) = 2V_{k,l}(i) + O(\delta_k^2)$.

IV. Algorithm 3: Stopping Time Problem

To motivate the algorithm, we introduce the stopping time problem and discuss how current policy evaluation algorithms are unsuitable as critic recursions for the SPSA-based policy-gradient methods proposed for all three algorithms. Consider a finite-state stopping time MDP where the actions applied in each state of the trajectory $\{X_0, X_1, ..., X_{\tau(\theta_k,i)} | X_0 = i\} \in S$ use policy $\theta_k = (\theta_k(j), j \in S)^T$, $k \geq 0$, stopping time $\tau(\theta_k, i)$ being a random variable. The cost-to-go is given by $V_k(i) = E\left(\sum_{n=0}^{\tau(\theta_k,i)-1} K(X_n, \theta_k(X_n)) | X_0 = i\right)$ for all $i \in S$. Due to the setting, $\theta_k(i) \in U_i \equiv \Pi_{m=1}^{N_i}[a_i(m), b_i(m)]$ for $a_i(m) < b_i(m)$ and, similar to the operators $P_{l,i}$ and P_l earlier (but without a subscript indicating stage), there exist projection operators P_i and P into corresponding U_i and U. For ease of exposition, assume that the policy iterate θ_k is not parametrized. Policy gradient methods used in [6], [9] and [7] would suggest a policy update iteration:

$$\theta_{k+1} \quad := \quad P(\theta_k - \frac{a_k}{2\delta_k}(V_k^+ - V_k^-) \circ \Delta_k^{-1}) \quad (5)$$

where $V_k^{\pm} = (V(\theta_k^{\pm}, j), \forall j \in S)^T$. Note that, in practice, noisy estimates of $V_k^{\pm}$ will be used.

Now assume that we parameterize the $V_k^{\pm}$ in (5): $V(\theta_k^{\pm}, i) = \langle\phi(i), r_k^{\pm}\rangle$, where $r_k^{\pm} \in \mathcal{R}^K$ for $K \ll |S|$, and $\phi(i)$ is the feature vector of state $i \in S$. Algorithms that apply the methods of TD-(λ) and its variant (cf. [2], and [1]) or LSTD-(λ) (cf. [4]) for policy evaluation are candidates to estimate the critic terms $V_k^{\pm}$. Such a policy evaluation has been applied to an autonomous stopping time problem (viz. a Markov Reward Process where only the stopping decision has to be taken in each state) in [3]. In contrast, here we have controls $\theta_k(i)$ in each state $i \in S$. When combined with the SPSA-type policy update (5), the above methods of policy evaluation have a potential flaw. Consider the $|S| \times K$ matrix $\Phi = (\phi(i), 1 \leq i \leq |S|)^T$ and $V_k^+ = V(\theta_k^+)$ indicating cost-to-go for perturbed policy iterate θ_k^+. Policy evaluation methods perform a projection of V_k^+ using a linear operator Π_{Φ,θ_k^+} to a point W in the span of Φ, minimizing a *weighted* Euclidean norm $\sum_{i=1}^{|S|} \pi_{\theta_k^+}(i)(V_k^+(i) - W(i))^2$, the weights $\pi_{\theta_k^+}$ being stationary probabilities of visiting state i under policy θ_k^+. In [2] there is a further bounded error w.r.t. W which is presently ignored. Thus, the projection operator depends on the policy θ_k^+ through the weights $\pi_{\theta_k^+}(i)$.

These concerns result in (5) being $\theta_{k+1} :=$

$$P(\theta_k - \frac{a_k}{2\delta_k}(\Pi_{\Phi,\theta_k^+}V_k^+ - \Pi_{\Phi,\theta_k^-}V_k^-) \circ \Delta_k^{-1}),$$

which performs gradient descent to a policy θ^* that minimizes $\Pi_{\Phi,\theta}V(\theta)$, not $V(\theta)$ component-wise. In contrast, the ideal analog of (5) would be:

$$\theta_{k+1} \quad := \quad P(\theta_k - \frac{a_k}{2\delta_k}(\Pi_\Phi V_k^+ - \Pi_\Phi V_k^-) \circ \Delta_k^{-1}),$$

where the projection operator Π_Φ is independent of the policy iterates θ_k.

The algorithm of Section II extends, and we do not provide a detailed description. The key difference is that, per iteration k, L trajectories from each $i \in S$ are simulated. Such trajectories - simulated until stopping time is hit - could be long, with differing stop times $\tau(\theta_k^+, i)$ and $\tau(\theta_k^-, i)$. Compare this with the algorithm of Section II that requires precisely $T-l$ length trajectories for the $l-$th stage, or of Section III that uses single transitions. Further, an analog of Assumption 1 is required here: $N_i = N$, $a_i(m) = a(m)$, and $b_i(m) = b(m)$, $\forall i \in S$.

V. Simulation Results

We consider a continuous time queuing model of flow control. This problem is related to Available Bit Rate (ABR) or Unspecified Bit Rate (UBR) explicit-rate flow control in ATM networks. An ATM source permits three broad categories of connections: ABR, UBR and Constant Bit Rate (CBR). ATM sources can infer that certain CBR connections will be established and will commence sending at a specific time-of-day, e.g. business connections begin at 9 AM and close by 6 PM. When a ABR or UBR connection establishment occurs outside these hours, the free bandwidth is available only for a finite horizon of time, i.e. until 9 AM the following day. The ATM source must clear the network by the time the CBR connections are established again. ATM Forum specifications do not require such flow-control, however tuning ABR rates with closed-loop feedback (e.g. using the proprietary Cisco ForeSight algorithm) help in better throughput at the ATM sources.

We assume that the source has a finite buffer of size $B < \infty$ and packets are fed by both an uncontrolled Poisson arrival stream (representing CBR) of rate $\lambda_u = 0.2$, and a controlled Poisson process (indicating ABR) of rate $\lambda_c(t)$ at instant $t > 0$. Service times are i.i.d., exponentially distributed with rate 2.0. The queue length process $\{X_t, t > 0\}$ at the node is observed every $\tilde{T}$ instants, for some $\tilde{T} > 0$, upto the instant $\tilde{T}T$, where T is the terminating stage. Say X_l denotes the queue length at instant $l\tilde{T}$, $0 \le l \le T$. The ABR connection is then signaled, and it starts sending packets at rate $\lambda_c(X_l)$ during the interval $[l\tilde{T}, (l+1)\tilde{T})$. We use $B = 1000$ and $T = 5$, and designate the 'target states' $\hat{T}_l$ as evenly spaced states within the queue i.e., $\hat{T}_l = B - \frac{l+1}{T+1}B$ that shift to 0 as the stage l increases from 0 to 5. Thus, $\hat{T}_1 = 666$, $\hat{T}_2 = 500$, $\hat{T}_3 = 333$, $\hat{T}_4 = 166$ and $\hat{T}_5 = 0$, respectively. The one-step transition cost under a given policy θ is taken as $K_l(X_l, \theta_l(X_l), X_{l+1}) = \left|X_{l+1} - \hat{T}_{l+1}\right|$ with $K_T(i) = 0, \forall i \in S$. This function penalizes deviations from the target states $\hat{T}_{l+1}$. The goal is to maximize throughput in the early stages (l small), while as l increases, the goal steadily shifts towards minimizing the queue length and hence the delay as one approaches T.

We used an order 3 polynomial approximation, implying storage of $K = 4$ coefficients. The feature vector $\phi_l(i) = (\phi_{l,k}(i), 1 \le k \le K)^T$ for each state i was:

Method	Time in sec.	Deviation	State
Approximate DP	3090	0	-
Algorithm 1	3923	31.67	532
Algorithm 2	2264	30.94	87

TABLE I: Performance of FH-MDP algorithms

Method	Time in sec.	Deviation	State
Value Iteration	3095	0	-
Algorithm 3	3500	138.49	956

TABLE II: Performance of Stopping-time algorithm

$$\phi_{l,k}(i) \quad = \quad \left(1.0 + \frac{i - \hat{T}_l}{\hat{T}_l}\right)^{k-1}. \qquad (6)$$

The averaging parameter L was 50, implying 50 starts from each (state, stage) pair. We used $\delta_k = k^{-0.167}$, $b_k = k^{-0.55}$ whilst the actor stepsize a_k was $(k\ln k)^{-1}$. We computed matrices $(\Phi_l\Phi_l)^{-1}$ needed in (1) before the algorithm commenced. To apply the DP algorithm, we discretized the interval $[0.05, 4.5]$ to obtain 25 equally spaced actions in each state. The transition matrix $P_{\tilde{T}}$ was computed using the method of [12, §6.8], which does not scale well in $|S|$ or $|U_{l,i}|$, hence the modest values of $|S| = 1000$ and $|U_{l,i}| = 25$. The computation time incurred in this matrix computation is included in the comparison of Table I. Note, however, that the proposed algorithms require far less memory, as $P_{\tilde{T}}$ storage is eliminated. A comparison of the cost-to-go from each state upon using the three policies is shown in Figure 2, these match each other for the most part. In Table I, 'Deviation' corresponds to maximum deviation from the costs-to-go of Approximate DP observed in Figure 2 and 'State' indicates the state at which this deviation was observed. All costs-to-go shown are obtained from 10^4 independent sample trajectories $\{X_0, X_1, ..., X_5\}$ for each initial state $X_0 = i$ and with a different initial seed. As seen in Figure 1, as the stage l increases, the optimal policy shifts to the left because of the form of the cost function. All code was run on a Pentium III computer and was written in the C programming language.

For the case of optimal control of stopping time processes, the algorithm proposed in Section IV is implemented on a finite buffer of size $B = 1000$. After starting from a state $X_0 = i$, the first time that the set of states $\{\lceil\frac{B}{3}\rceil, \lceil\frac{B}{3}\rceil + 1, ..., \lceil\frac{2B}{3}\rceil\}$ of this buffer is hit by the queue-length process X_t is the stopping time $\tau(\theta_n, i)$, for a system operating under policy θ_n. The cost incurred until $\tau(\theta_n, i)$ is the sum of the one-step transition costs $K(X_t, \theta_n(X_t), X_{t+1}) = \left|X_{t+1} - \frac{B}{2}\right|$ where $0 \le t \le \tau(\theta_n, i)$. The feature vectors $\phi_k(i)$ used are just as (6), $\forall i \in S$ and $1 \le k \le 4$, with the modification that $\hat{T}_l = \frac{B}{2}$, $\forall 1 \le l \le T$. To speed up the first hit-time, we used $\mu = 2.0$ as a lower bound for states $i \le \frac{B}{2}$ during the execution of the algorithm, meaning that $\lambda_c \in [2.0, 4.5]$. Similarly, we used 2.0 as the upper bound for states $i > \frac{B}{2}$ resulting in $U_i = [0.05, 2.0]$ for such states. Figure 3 shows the policy computed using the value iteration (VI) algorithm where knowledge of the transition probabilities is required. The VI algorithm converged to optimal $V(\theta^*, i)$, $\forall i \in S$ in 50 iterations, to which we add the time required to compute the $P_{\tilde{T}}$ transition matrices. The total cost is compared in Figure 4, the costs-to-go for both policies being very similar, whilst performance is compared in Table II.

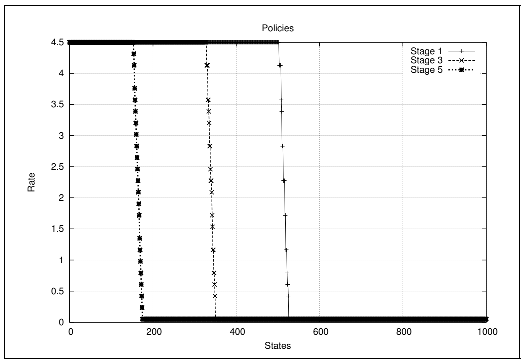

Fig. 1: Optimal Policy using Approximate DP

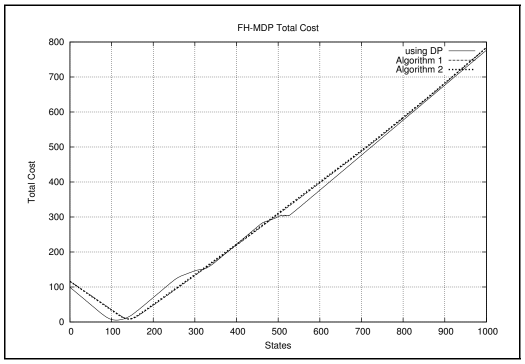

Fig. 2: Comparison of Finite-Horizon costs-to-go

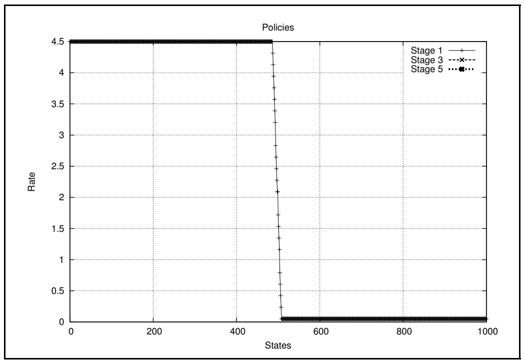

Fig. 3: Optimal Policy using value iteration

VI. Conclusions

We proposed two parameterized actor-critic algorithms for F-H MDPs. The parameterization requires constant storage: viz. $\left(2K + K^2\right) \times T$, where $K \ll |S|$, although the more efficient second algorithm needs additional $2|S|$ storage. This contrasts with [6], where at least $2|S| \times T$ is needed, even assuming there are only two feasible actions per state. On the other hand, the proposed algorithms have time requirements $O(0.5T^2)$ and $O(3T)$ compared to $O(2T)$ or $O(T)$ in [6]. An extension of the algorithm to the stopping time problem was discussed in Section IV, and we identified why current policy evaluation schemes do not produce critic

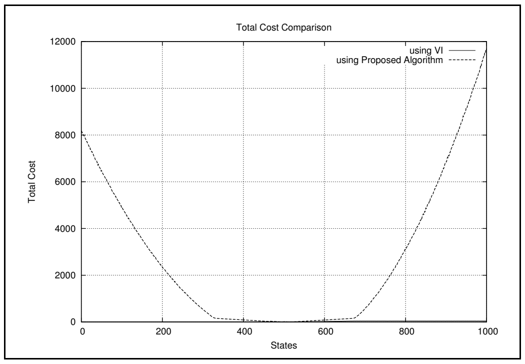

Fig. 4: Comparison of costs till stopping-time

estimates suitable for an SPSA policy-gradient method. One solution to the high simulation load in Algorithm 1 lies in using one-simulation SPSA estimates, a method adopted (for the look-up table case) in [9]. Also, algorithms currently under investigation aim to reduce the variance encountered in the one-simulation SPSA gradient estimate by re-using past estimates of $V^{+}_{k,l}$.

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] J. Tsitsiklis and B. Van Roy, “Average Cost Temporal–Difference Learning,” *Automatica*, vol. 35, no. 11, pp. 1799–1808, 1999.

[2] ——, “An Analysis of Temporal–Difference Learning with Function Approximation,” *IEEE Transactions on Automatic Control*, vol. 42, no. 5, pp. 674–690, 1997.

[3] ——, “Optimal Stopping of Markov Processes: Hilbert Space Theory, Approximation Algorithms, and an Application to Pricing High-Dimensional Financial Derivatives,” *IEEE Transactions on Automatic Control*, vol. 44, no. 10, pp. 1840–1851, 1999.

[4] J. Boyan, “Least–squares Temporal Difference Learning,” in *Proceedings of the Sixteenth International Conference (ICML) on Machine Learning*, 1999, pp. 49–56.

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

[6] S. Bhatnagar and M. Abdulla, “Reinforcement learning based algorithms for finite horizon markov decision processes,” *Submitted*, 2005.

[7] M. Abdulla and S. Bhatnagar, “Reinforcement learning based algorithms for average cost markov decision processes,” *Accepted for publication in Discrete Event Dynamical Systems*, 2006.

[8] P. Marbach and J. Tsitsiklis, “Simulation-Based Optimization of Markov Reward Processes,” *IEEE Transactions on Automatic Control*, vol. 46, no. 2, pp. 191–209, 2001.

[9] S. Bhatnagar and M. S. Abdulla, “An actor-critic algorithm for finite horizon markov decision processes,” in *Proceedings of the 45th IEEE-CDC, Dec 11-13, San Diego, CA, USA*, 2006.

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

[11] H. Kushner and D. Clark, *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. New York: Springer Verlag, 1978.

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