SOLUTION OF MDPS USING SIMULATION-BASED VALUE ITERATION

Mohammed Shahid Abdulla and Shalabh Bhatnagar
Department of Computer Science and Automation, Indian Institute of Science, Bangalore – 560 012, India.

Abstract: This article proposes a three-timescale simulation based algorithm for solution of infinite horizon Markov Decision Processes (MDPs). We assume a finite state space and discounted cost criterion and adopt the value iteration approach. An approximation of the Dynamic Programming operator *T* is applied to the value function iterates. This `approximate' operator is implemented using three timescales, the slowest of which updates the value function iterates. On the middle timescale we perform a gradient search over the feasible action set of each state using Simultaneous Perturbation Stochastic Approximation (SPSA) gradient estimates, thus finding the minimizing action in *T*. On the fastest timescale, the `critic' estimates, over which the gradient search is performed, are obtained. A sketch of convergence explaining the dynamics of the algorithm using associated ODEs is also presented. Numerical experiments on rate based flow control on a bottleneck node using a continuous-time queueing model are performed using the proposed algorithm. The results obtained are verified against classical value iteration where the feasible set is suitably discretized. Over such a discretized setting, a variant of the algorithm of [12] is compared and the proposed algorithm is found to converge faster.

FRAMEWORK AND INTRODUCTION

We consider an MDP $\{X_t, t = 0,1,...\}$ where decisions are made at instants $t = 0,1,...$ using an associated control-valued process $\{Z_t\}$. Suppose $S \equiv \{1,2,...s\}$ is the (finite) state space and C is the control space. Suppose also that $U(i) \subseteq C$ is the set of all feasible controls in state i. Let $p(i,u,j), i,j \in S, u \in U(i)$ be the transition probabilities associated with this MDP. An admissible policy $\mu = \{\mu^0, \mu^1, \mu^2, ...\}$ with

$\mu^t : S \to C, t \geq 0$, is such that $\mu_i^t \in U(i)$. An admissible policy μ is stationary if $\mu^t = \gamma, \forall t$. Suppose $K(i_t, u_t, i_{t+1})$ denotes the one-step transition cost when the current state is i_t, the action chosen is $u_t \in U(i_t)$ and the next state is i_{t+1}. The aim here is to find a stationary admissible policy π that minimizes for each state $i \in S$ the associated infinite horizon discounted cost $V_\pi(i)$, where

$$V_\pi(i) = E\{\sum_{t=0}^{\infty} \alpha^t K(i_t, \pi_{i_t}, i_{t+1}) \mid i_0 = i\} \tag{1}$$

and $0 < \alpha < 1$ is the discount factor. For any state $i \in S$, the minimum infinite-horizon discounted-cost V_i^* satisfies the *Bellman equation*

$$V_i^* = \min_{u \in U(i)} E\{K(i,u,\eta(i,u)) + \alpha V^*_{\eta(i,u)}\} \tag{2}$$

the random variable $\eta(i,u) \in S$ being the state to which the system transits upon application of control u in state i. The expectation above is over $\eta(i,u)$. To compute $V^* = (V_i^*, \forall i \in S)^T$ we may apply the Dynamic Programming algorithm recursively:

$$V_i(n+1) = \min_{u \in U(i)} E\{K(i,u,\eta(i,u)) + \alpha V_{\eta(i,u)}(n)\} \tag{3}$$

Note that here $V(n) = (V_i(n), \forall i \in S)^T$ are iterates unlike defined quantities V_π in (1). It is known that as $n \to \infty$, the iterates $V_i(n)$ converge to V_i^* exponentially. We may rewrite (3) as $V_i(n+1) = T_i(V(n))$ where $T : \Re^s \to \Re^s$ such that $T = (T_i, \forall i \in S)^T$ is the Dynamic Programming (DP) operator and the vector V^* is the unique fixed point of T . This method of continually applying T is termed as successive approximation or value iteration (cf. sec. 8.2 of [2]). The optimal policy π^* (where $V_{\pi^*} = V^*$) can be inferred from (2), in general, provided the system transition probabilities $p(i, \pi_i^*, \cdot)$ are known, which is not true in most real-life applications. When $p(i, \pi_i, \cdot)$ are available but not obtainable via a closed form expression, storing these values when $|S|$ is large is difficult. Even so, such a possibility is ruled out in the case where $U(i)$ are compact or countably infinite. Denote $\pi_i(n)$ as the 'greedy' policies w.r.t. the iterates $V(n)$ of (3), i.e.

$$\pi_i(n+1) = \arg\min_{u \in U(i)} E\{K(i,u,\eta(i,u)) + \alpha V_{\eta(i,u)}(n)\} \tag{4}$$

then we may infer that $\pi_i(n) \to \pi_i^*$ although tight bounds of the form $V_i(n) + \underline{c}_i(n) \leq V_i^* \leq V_i(n) + \underline{c}_i(n)$, where $\underline{c}_i(n) \leq \underline{c}_i(n)$, available in value iteration hold for $\pi_i(n)$ if $\pi_i(n)$ belong to a space with a well-defined metric topology, e.g. $\pi_i(n) \in \Re^s, \forall n \geq 0$. Therefore, we assume a compact action set $U(i) \in \Re^{N_i}, \forall i \in S$.

Simulation based approaches to value iteration in the literature go under the rubric *critic-only* algorithms. See, for example, sec. 1 of [9], where the drawbacks of such schemes are also identified. In a typical simulation based implementation of T_i, first the estimation of $E\{K(i,u,\eta(i,u)) + \alpha V_{\eta(i,u)}(n)\}$ is performed, for each action $u \in U(i)$. This procedure is not feasible when $U(i)$ is a compact or even a countably infinite set. This is followed by finding the minimum of such terms over $U(i)$, we call this minimum $\widetilde{V}_i(n+1)$, and (possibly) storing the minimizing u as iterate $\widetilde{\pi}_i(n+1)$. Note that this minimization step has computational complexity $O(\sum_{i=1}^{s} |U(i)|)$. An example of critic-only value iteration are the algorithms in sec. 6.2 and 6.3 of [12], that employ online TD(0) learning, both without and with exploration of $U(i)$, respectively. However, these algorithms use value function approximation whereas we target the look-up table case. An important difference in approach is that in [12], the system is simulated using a single endless trajectory to enable learning of the coefficients involved in value function approximation. As we shall show in sec. 4, this reduces to simulating single transitions out of each state $i \in S$ repeatedly. Further, the setting there is also somewhat different in that one does not have to use a simulator in order to estimate the minimizing $\widetilde{\pi}_i(n+1)$. As a result, it is easy to find the resultant greedy policy $\widetilde{\pi}_i(n+1)$ once the value function iterate $V_i(n)$ is identified. This property of such systems is what makes value iteration attractive to implement. Two examples of other systems where this property holds are to be found in [11] and [10].

In contrast, we use the actor-critic approach to approximate T_i. Gradient search is performed using a slower timescale recursion for finding the minimizing control in (3). Thus, given value function estimates $V(n)$, the iterates $\widetilde{\pi}(n+1)$ represent an approximation to $\pi(n+1)$ of (4). To perform gradient descent we employ SPSA estimates of the true gradient $\nabla_u E\{K(i,u,\eta(i,u)) + \alpha V_{\eta(i,u)}(n)\}$ in a stochastic approximation recursion, similar to the policy gradient approach of [4] and [1]. Note that the difficulty here arises since it is not only the expectation $E\{\cdot\}$ that needs to be evaluated but also the value function itself, before the gradient is estimated. Clearly, the objective function here does not possess any analytically obtainable expression and hence needs to be estimated. In particular, to obtain control $\widetilde{\pi}_i(n+1)$ we evaluate $E\{K(i,u^r,\eta(i,u^r)) + \alpha V_{\eta(i,u^r)}(n)\}$, $r = 1,2$ at two 'perturbed' actions, u^1 and u^2 that will be made precise later. A faster time-scale recursion estimates the two $E\{\cdot\}$ terms required above by independently simulating M transitions from state i using actions u^1 and u^2, respectively, thus permitting two 'parallel' simulations. Note that in the normal course, given a control $u \in U(i)$, one would wait for convergence of iterates towards the $E\{\cdot\}$ terms required above. Instead we only perform a fixed M steps of averaging the random variables $E\{K(i,u^r,\eta^r_m(i,u^r)) + \alpha V_{\eta^r_m(i,u^r)}(n)\}, 0 \le m \le M-1, r = 1,2$, where η^r_m now stands for the two parallel M-element random processes having the law $p(i,u^r,\cdot)$. Note that in the above, the convergence analysis permits us to fix M at 1 - although better performance was obtained when a larger M was used. Convergence in the proposed algorithm is achieved because of the different step size schedules or timescales as with [1], [3], [4], [5], [8], and [9]. While in critic-only methods the resultant optimal policies are not explicitly computed, a highlight of the proposed algorithm is that due to its actor-critic structure, the convergence criteria can now be designed based on *both* the policy iterates $\tilde{\pi}$ and the value-function iterates V. This helps to accommodate problem-specific sensitivity to these quantities. Such a feature is achieved due to both these quantities being estimated in the algorithm.

ALGORITHMS

Suppose $U(i), i \in S$ are of the form $U(i) = \prod_{j=1}^{N_i} [a_i^j, b_i^j]$, where $a_i^j, b_i^j \in \Re$. We make the assumption that $\forall i,j \in S, u \in U(i)$, both $K(i,u,j)$ and $p(i,u,j)$ are continuously differentiable in u.

Let $P_i^j(y) = \min(b_i^j, \max(a_i^j, y)), y \in \Re$ be the projection of y onto the interval $[a_i^j, b_i^j], 1 \le j \le N_i$. Further, for $y = (y_1, y_2, \ldots, y_{N_i})^T \in \Re^{N_i}$, let $P_i(y) = (P_i^1(y_1), P_i^2(y_2), \ldots, P_i^{N_i}(y_{N_i}))^T$. Then $P_i(y)$ denotes the projection of $y \in \Re^{N_i}$ to the set $U(i), i \in S$. Also define an operator $P(\cdot)$ as $P(x) = (P_1(x_1), P_2(x_2), \ldots, P_s(x_s))^T$ where $x_j \in \Re^{N_j}, 1 \le j \le s$, and $x = (x_1, x_2, \ldots, x_s)^T$.

The recursion that tracks (3) is the following: For all $i \in S$ and $n \ge 0$:

$$V_i(n+1) = V_i(n) + a(n)(\tilde{V}_i(n+1) - V_i(n)) \tag{5}$$

where $\tilde{V}_i(n+1) = \left(\tilde{V}^1{}_i((n+1)M) + \tilde{V}^2{}_i((n+1)M)\right)/2$ is an approximation to $T_i(V(n))$ and $a(n)$ is a diminishing step size. Here, M is the number of instants over which additional averaging in recursion (7) is performed. The iterate $\tilde{V}$ will be made precise below in the description of (7). Recursion (6) updates $\tilde{\pi}(\cdot)$:

$$\tilde{\pi}_i(n+1) = P_i(\tilde{\pi}_i(n) + b(n)\frac{\tilde{V}_i^1((n+1)M) - \tilde{V}_i^2((n+1)M)}{2\delta\Delta_i(n)}) \tag{6}$$

Here, $\Delta_i(n)$ are deterministic normalized Hadamard-matrix based perturbations generated according to the method in sec. 2.2.1 of [3] (see also [5] where Hadamard matrix based perturbations were first introduced) whereas $\delta \geq 0$ is a small constant. To produce the iterates $\tilde{V}_i^1((n+1)M)$ and $\tilde{V}_i^2((n+1)M)$ consider perturbed policies $\tilde{\pi}_i^1(n) = P_i(\tilde{\pi}_i(n) - \delta\Delta_i(n))$ and $\tilde{\pi}_i^2(n) = P_i(\tilde{\pi}_i(n) + \delta\Delta_i(n))$. Now perform the following iterations for $r = 1,2$ and $m = 0,1,\dots,M-1$:

$$\tilde{V}_i^r(nM+m+1) = \tilde{V}_i^r(nM+m) + c(n)\begin{pmatrix} K(i,\tilde{\pi}_i^r(n),\eta_{nM+m}^r(i,\tilde{\pi}_i^r(n))) \\ +\alpha V^r_{\eta_{nM+m}^r(i,\tilde{\pi}_i^r(n))}(nM+m) \\ -\tilde{V}_i^r(nM+m) \end{pmatrix} \tag{7}$$

Here, we run two simulations corresponding to $r=1$ and $r=2$ in parallel. Note that in the above, η^1 and η^2 represent two stochastic processes with elements of the form $\eta_k^r(i,u), k \geq 0$ which are $S-$valued random variables, independently generated using $p(i,u,\cdot)$. The requirements on the step-sizes $a(n)$, $b(n)$ and $c(n)$ used in (5), (6) and (7), respectively, are as follows: $d(n) > 0$, $\sum_n d(n) = \infty$, $\sum_n d^2(n) < \infty$ where $d(n)$ represents all three sequences $a(n), b(n)$ and $c(n)$, alongwith:

$$a(n) = o(b(n)), b(n) = o(c(n)). \tag{8}$$

Thus $a(n) \to 0$ is the fastest and $c(n) \to 0$ is the slowest among the three step-size schedules. As a result, the timescale corresponding to $\{a(n)\}$ is the slowest while that corresponding to $\{c(n)\}$ is the fastest. This is because beyond some finite $N_0 > 0$ (i.e., for $n > N_0$), the increments in the recursion governed by $\{a(n)\}$ are uniformly the smallest while those in recursion governed by $\{c(n)\}$ are uniformly the largest among the three. Hence recursions governed by $\{c(n)\}$ converge the fastest (even though they exhibit higher variability in their iterates) amongst the three recursions. Note that we could replace (5) with the direct-assignment recursion $V_i(n+1) = \tilde{V}_i(n+1)$. However, using stochastic averaging with step-size $a(n)$ results in graceful convergence properties in our experiments. Such a choice between the averaging of (5) and direct-assignment is an example of the *bias-variance* dilemma.

OUTLINE OF PROOF

We explain briefly the intuition behind the algorithm, using the Ordinary Differential Equation (ODE) approach for easy illustration and only supply pointers to related proofs already in the literature.

Suppose we define the $\Re-$valued function $\overline{V}_i(W,u)$ (for any $W \in \Re^s$ and $u \in U(i)$) as follows: $\overline{V}_i(W,u) = E\{K(i,u,\eta(i,u)) + \alpha W_{\eta(i,u)}\}$ .It is easy to see that, for a given stationary admissible policy π, the vector function $\overline{V}(W,\pi) = (\overline{V}_1(W,\pi_1), \overline{V}_2(W,\pi_2),\ldots,\overline{V}_s(W,\pi_s))^T$ is in fact the appropriate constrained DP operator T_π. Also, (3) may be written as $V_i(n+1) = T_i(V(n)) = \min_{u\in U(i)}\{\overline{V}_i(V(n),u)\}$. Recall that the fixed point of the operator T_π, $V_\pi = T_\pi(V_\pi)$ is the solution of the Poisson Equation: $V_\pi(i) = E\{K(i,\pi_i,\eta(i,\pi_i)) + \alpha V_\pi(\eta(i,\pi_i))\}$, where $V_\pi(k)$ represents the $k-$th element of the vector V_π.

Using a proof technique resembling that of Corollary 4.2 of [4], for any $n \geq 0$ the asymptotically stable ODE that the faster recursion (7) tracks is, for $t \geq 0$

$$\dot{\widetilde{V}}_i^r(t) = \overline{V}(V(t), \pi_i^r(t)) - \widetilde{V}_i^r(t) \tag{9}$$

Note that the terms $\widetilde{\pi}_i^r(t)$ and $V(t)$ are quasi-static at the time scale corresponding to (7) as recursions (6) and (5) proceed on slower timescales (cf. (8)). Hence we may replace $\pi_i^r(t)$ with π_i^r and $V(t)$ with V, respectively. The asymptotically stable equilibrium point of (9) would be $\widetilde{V}_i^r = \overline{V}(V, \widetilde{\pi}_i^r)$. By virtue of (8) again, (7) would be seen by (6) (and (5)) as having converged. Using the iterates $\widetilde{V}_i^1((n+1)M)$, a better estimate $\widetilde{\pi}(n+1)$ to the policy $\pi(n+1)$ of (4) will be produced via the SPSA update rule of (6).

For the time being, we assume that the SPSA based estimate in (6) performs correct gradient descent towards $\pi_i(n+1)$ (the proof technique for such a claim resembles that of Lemma 4.10 of [4]). This implies that (6) tracks the asymptotically stable (projected) ODE:

$$\dot{\pi}_i(t) = \hat{P}_i(-\nabla_u \overline{V}_i(V,u)), \tag{10}$$

whose equilibrium point is the desired minimum. Here, $\hat{P}_i = (\hat{P}_i^j, 1 \leq j \leq N_i)^T$ and $\hat{P}_i^j(v(y)) = \lim_{\eta\to 0}\left(P_i^j(y+\eta v(y)) - y\right)/\eta$ for any bounded, continuous and real-valued function v. Further, (5) views recursions (6) and (7) as having converged and so considers an estimate to

$V_i(n+1) = T_i(V(n))$ as available at its $(n+1)-$ th update. Note that recursion (5) has no direct use for the $\widetilde{V}_i^r((n+1)M)$ terms of (7) and instead requires an approximation to $T_i(V(n))$. However, we can employ the iterates $\widetilde{V}_i^r$ in (5), as we describe below. The novelty is that this role of iterates $\widetilde{V}^r$ is in addition to the role played as 'critic' for recursion (6).

Note that iterate $\widetilde{\pi}(n+1)$ does approximate $\pi(n+1)$ (by virtue of (10)), and we may use $\overline{V}(V(n), \widetilde{\pi}(n+1))$ as $\widetilde{V}_i(n+1)$ in (5). However, we would need additional simulation to estimate the former term. This we can avoid by considering the quantities $\overline{V}_i(V(n), \widetilde{\pi}_i^r(n))$, for $r = 1,2$, approximated by the iterates $\widetilde{V}_i^r((n+1)M)$. Using Taylor series expansion (as in the proof of Lemma 4.10 of [4]), we see that $V_{\widetilde{\pi}^1(n)}(i) + V_{\widetilde{\pi}^2(n)}(i) = 2V_{\widetilde{\pi}(n)}(i) + O(\delta)$ and use this fact in (5). Note that no properties of the perturbations Δ_i are used in the above equality. Further, it is an estimate of $\overline{V}_i(V(n), \widetilde{\pi}_i(n))$ that is available to us and not the more recent $\overline{V}_i(V(n), \widetilde{\pi}_i(n+1))$. This represents the loss in accuracy due to *not* estimating $\overline{V}_i(V(n), \widetilde{\pi}(n+1))$ by simulation.

NUMERICAL RESULTS

We consider a continuous time queuing model for flow control in communication networks as in [4]. A single bottleneck node is fed with two arrival streams, one an uncontrolled Poisson stream and the other a controlled Poisson process. Service times are assumed i.i.d., with exponential distribution. We assume that the node has a buffer size $B < \infty$. Given a constant $T > 0$, we assume that the continuous-time queue length process $\{q_t, t > 0\}$ at the node is observed every T instants and on the basis of this information the controlled source tunes the rate at which it sends packets so as to minimize a certain cost. Suppose q_n denotes the queue length observed at time $nT, n \geq 0$. The controlled source thus sends packets according to a Poisson process with rate $\lambda_c(q_n)$ during the time interval $[nT, (n+1)T]$. The rate is then changed to $\lambda_c(q_{n+1})$ at instant $(n+1)T$ upon observation of state q_{n+1} - we assume for the present that there is no feedback delay. The aim then is to find a stationary optimal rate allocation policy that minimizes the associated infinite horizon discounted cost. We choose $B = 50$ and take the compact action set $U(i)$ (where $\lambda_c(i) \in U(i), \forall i \in S$) to be the interval $[0.05, 4.5]$. The discount factor α is 0.9.

The uncontrolled traffic rate λ_u is chosen as 0.2 and the service rate μ for incoming packets is set to 2.0. For a system operating under a stationary policy π, we use the cost function $K(q_n, \pi_{q_n}, q_{n+1}) = |q_{n+1} - B/2|$ Note that while the source rate chosen, $\lambda_c(q_n) \equiv \pi_{q_n}$, does not directly enter into the cost function above, it has an impact on the queue length q_{n+1} observed T seconds later, which in turn affects the cost. A cost function of this type is useful in cases where the goal is to simultaneously maximize throughput and minimize the delay in the system.

The value of δ needed in (6) is set to 0.1. We arbitrarily set the initial value function iterate $V_i(0) = 0, \forall i \in S$ and policy iterate as $\tilde{\pi}_i(0) = 2.275, \forall i \in S$ for the proposed algorithm. The value of M needed in (7) is taken to be 100. As a metric to characterize the convergence of all algorithms, we measured a quantity $err_V(n)$ at value function update n in the following manner: $err_V(n) = \max_{i \in S, 1 \le k \le 50} \{|V_i(n) - V_i(n-k)|\}$. Similarly, convergence in terms of policy is obtained using $err_{\tilde{\pi}}(n) = \max_{i \in S, 1 \le k \le 50} \{\tilde{\pi}_i(n) - \tilde{\pi}_i(n-k)\}$ The step-size sequences $\{a(n)\}$, $\{b(n)\}$ and $\{c(n)\}$ needed in (5), (6) and (7), respectively, were chosen as $a(0) = b(0) = c(0) = 1$, $a(n) = n^{-1}, b(n) = n^{-0.9}, c(n) = n^{-0.55}$ respectively.

Convergence of the form $err_V(n) \le 0.1$ and $err_{\tilde{\pi}}(n) \le 0.1$ is achieved within 150 and 300 updates of (5), respectively. In Figure 1a, we plot the converged source rates for $T = 5, 10$ and 15 respectively. Similarly, converged value functions (using the err_V criterion) are shown in Figure 1b. We observe that for given T, the source rates are inversely related to the queue length values since the cost function imposes a high penalty for states away from $B/2$. Further, the difference between the highest and lowest rates decreases as T increases since for lower values of T, the controller has better control over the system dynamics than when T is large.

Convergence for the classical value iteration algorithm (3) for $err_V(n) \le 0.1$, is achieved within 100 $V(n)$ updates for all T. The value functions obtained therein are shown in Figure 2a, which match the corresponding results of the proposed algorithm shown in Figure 1b. We made some modifications to $U(i)$ in order to compute the transition probabilities $p(i,u,j), \forall i,j \in S, u \in U(i)$ needed in the RHS of the Poisson Equation $\sum_{j=1}^{S} p(i,u,j)(K(i,u,j) + \alpha V_j(n))$. The discrete action set $U(i), \forall i \in S$, consists of actions spaced at roughly 0.005 within the compact interval $[0.05, 4.5]$, thus making for $|U(i)| = 225, \forall i \in S$. The transition probabilities were computed in a method similar to that in [3]. On a Pentium 4 computer, classical value iteration (for the above discretized set) took 2 seconds whereas the proposed algorithm took 12 seconds (over the compact action set). This is due to the time required for simulating transitions, and a faster simulator will bring such a comparison closer.

For purposes of comparison, we propose a variant of the algorithm in sec. 6.2 and 6.3 of [12] for the current situation of a look-up table. We first present the algorithm and then its proposed variant to suit our framework. The algorithm proposed in [12] simulates a single endless trajectory, i.e. $i_{n+1} = \eta_n(i_n, u_n(i_n))$ and updates a coefficient vector that helps approximate the value function for each state $i \in S$. This coefficient vector is $r(n) \in \Re^K$ where $K << s$, and the approximate value function is given by $\widetilde{V}_i(n) = (\Phi r(n))(i), \forall i \in S$. Here, the $s \times K$ matrix Φ consists of the feature vectors $\phi(i), \forall i \in S$ as the rows and is of full rank. The $i-$th element of the $s \times 1$ vector $\Phi r(n)$ is indicated by $(\Phi r(n))(i)$. Below is the algorithm of sec. 6.2 of [12]:

$$r(n+1) = r(n) + a(n)\phi(i_n)\begin{pmatrix} K(i_n, u_n(i_n), i_{n+1}) + \alpha(\Phi r(n))(i_{n+1}) \\ -(\Phi r(n))(i_n) \end{pmatrix}, \quad (11)$$

where the decision $u_n(i_n)$ is the `greedy' decision, i.e.,

$$u_n(i_n) = \arg\min_{u \in U(i_n)} E\{K(i_n, u, \eta(i_n, u)) + \alpha(\Phi r(n))(\eta(i_n, u))\}. \quad (12)$$

Now, consider the value function iterates $V(n)$ in place of the coefficient vector $r(n)$ in (11). Also make the feature vector $\phi(i), \forall i \in S$ equal to the unit vector $e_i \in \Re^s$, then the algorithm (11) reduces to:

$$V(n+1) = V(n) + a(n)\begin{pmatrix} \sum_{i=1}^{s} \{K(i, u_n(i), \eta(i, u_n(i))) + \alpha V_{\eta(i, u_n(i))}(n)\} e_i \\ -V(n) \end{pmatrix}, \quad (13)$$

where recall that $u_n(i)$ is just the $\pi_i(n+1)$ in (4). Note that whereas (13) requires $u_n(i)$ for all $i \in S$, (11) uses only $u_n(i_n)$. Using the fact that $\Phi^T \Phi$ is the identity matrix, one can see that (13) is a variant for the more recent algorithm for approximate fixed point computation given in [6] as well, which employs a Kalman filter approach. In most simulation based settings, the term $u_n(i)$ can only be identified by simulating transitions out of i using all $u \in U(i)$, possibly averaging over multiple transitions for the same action $u \in U(i)$. Note that (13) corresponds to a stochastic approximation implementation of the operator T, using the diminishing stepsize $a(n)$. In [12], sec. 6.3 suggests that a greedy selection like (12) does not work in practice and exploration of $U(i)$ is employed. Further, [7] theoretically formalizes this using a T_δ operator which employs a Boltzmann distribution to select $u_n(i_n)$ from $U(i_n)$. We implement the variant of the algorithm of [12], using the exploration suggested in [7], and discretizing each $U(i)$ into 45 equally spaced actions. The algorithm uses $a(n) = n^{-0.6}$ and requires roughly 4000 updates and almost 800 seconds in all cases of T for the $V(n)$ vector to converge to $err_V(n) \leq 0.1$. Figure 2b plots the value function iterates obtained after convergence.

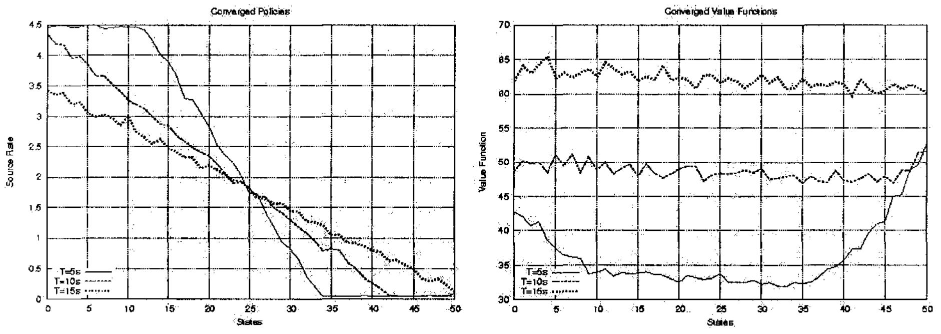

Figure 1: Plots of (a) converged policies and (b) value functions

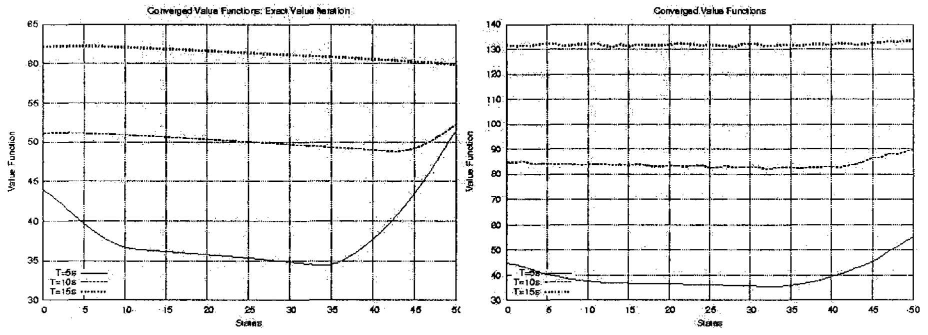

Figure 2: V iterates obtained using a) classical VI b) algorithm in [12]

ACKNOWLEDGEMENTS

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

REFERENCES

[1] Abdulla, M.S., and Bhatnagar, S. Reinforcement Learning Based Algorithms for Average Cost Markov Decision Processes. *Submitted.*

[2] Bertsekas, D.P. *Dynamic Programming and Stochastic Control,* 1976 New York: Academic Press.

[3] Bhatnagar, S., and Abdulla, M.S. Reinforcement Learning Based Algorithms for Finite Horizon Markov Decision Processes. *Submitted.*

[4] Bhatnagar, S., and Kumar, S. A Simultaneous Perturbation Stochastic Approximation-Based Actor-Critic Algorithm for Markov Decision Processes. IEEE Trans. on Automatic Control, 2004, 49(4):592~598.

[5] Bhatnagar, S., et al., Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences. ACM Trans. on Modeling and Computer Simulation, 2003, 13(4):180~209.

[6] Choi, D. S., and Van Roy, B. A Generalized Kalman Filter for Fixed Point Approximation and Efficient Temporal-Difference Learning, *Submitted to Discrete Event Dynamic Systems.*

[7] De Farias, D.P., and Van Roy, B. On the Existence of Fixed Points for Approximate Value Iteration and Temporal-Difference Learning. Journal of Optimization Theory and Applications, June 2000, 105(3).

[8] Konda, V.R., and Borkar, V.S. Actor-Critic Type Learning Algorithms for Markov Decision Processes. SIAM J. Control Optim., 1999, 38(1): 94~123.

[9] Konda, V.R., and Tsitsiklis, J.N. Actor-Critic Algorithms. SIAM J. Control Optim., 2003, 42(4): 1143~1166.

[10] Singh, S., and Bertsekas, D. Reinforcement Learning for Dynamic Channel Allocation in Cellular Telephone Systems. Advances in Neural Information Processing Systems (NIPS), 1997, 9: 974~980.

[11] Tsitsiklis, J. N., and Van Roy, B. Optimal Stopping of Markov Processes: Hilbert Space Theory, Approximation Algorithms, and an Application to Pricing High-Dimensional Financial Derivatives. IEEE Trans. on Automatic Control, 1999, 44(10): 1840~1851.

[12] Van Roy, B., et. al., A Neuro-Dynamic Programming Approach to Retailer Inventory Management, 1997, Proc. of the IEEE Conf. on Decision and Control..