Sādhanā, Vol. 22, Part 4, August 1997, pp. 525–543. © Printed in India.

The actor–critic algorithm as multi-time-scale stochastic approximation

VIVEK S BORKAR* and VIJAYMOHAN R KONDA

Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560 012, India

Abstract. The actor–critic algorithm of Barto and others for simulation-based optimization of Markov decision processes is cast as a two time scale stochastic approximation. Convergence analysis, approximation issues and an example are studied.

Keywords. Actor-critic algorithm; stochastic approximation; Markov decision processes; simulation-based algorithms; policy iteration.

1. Introduction

Markov Decision Processes (MDPs) have been a popular paradigm for sequential decision making under uncertainty. The traditional approach has been to write down the dynamic programming equations appropriate for the problem at hand. Their solution yields the so-called value function for the problem. The optimal policy as a function of the state is then prescribed as the minimizer of a certain 'Hamiltonian' defined in terms of the value function. This, in fact, gives a complete characterization of optimal (stationary) policies. Since everything hinges on computation of the value function, several iterative algorithms have been proposed for the same. These fall broadly into two classes : value iteration and policy iteration. An extensive account of these and related developments can be found in Puterman (1994).

When applied to real problems, however, this scheme often runs into difficulties. The most notorious, of course, is the curse of dimensionality, caused by the typically very large size of the state space. An equally (if not more) difficult issue stems from the fact that the theoretical analysis of MDPs presupposes exact knowledge of underlying stochastic dynamics. Translated into real terms, this calls for accurate model selection and identification of the relevant parameters. Though this can in principle be a separate, off-line statistical exercise, the computational overheads can be considerable.

This problem has been brought to the fore forcefully by some emerging applications in artificial intelligence, e.g., in game playing machines and robotics (Barto *et al* 1995; Keerthi & Ravindran 1994). Here complexity of exact modeling and analysis is very high,

*Author for correspondence

but, mercifully, that of simulation is often not so. To press this point further, consider a large interconnected system like a communication network. The dynamics evolves as per simple local update rules operating at individual constituent units and therefore is quite amenable to simulation on a parallel machine. But the overall dynamics can be extremely hard to analyse. This has prompted simulation-based algorithms which are akin to the traditional algorithms for computing the value function, but with one crucial difference: one replaces a computation using an exactly known transition probability function by an actual simulated transition as per the random mechanism that determines it. The algorithm is expected to 'see' the actual transition mechanism through an averaging affect. Mathematically speaking, these involve a marriage between the traditional algorithms for MDPs and stochastic approximation, a procedure in statistics with a long and illustrious history (see Benveniste *et al* 1990, for a comprehensive account of the latter).

These algorithms again fall into two broad classes. The first is the Q-learning algorithm of Watkins (1989) which has been extensively analysed (Watkins & Dayan 1992; Jaakola *et al* 1994; Tsitsiklis 1994). This can be viewed as a stochastic approximation version of the value iteration. The other strand consists of the actor–critic algorithm of Barto *et al* (1983) and its variants which may be viewed as stochastic approximation counterpart of policy iteration. Though some mathematical analysis is available in this case as well (Williams & Baird 1990), the situation is a little harder than Q-learning because of a certain inherent problem in policy iteration. In policy iteration, each update requires the computation of 'cost to go' function or 'value' function for a fixed policy. This in itself requires a 'value iteration' sandwiched between two policy updates, albeit a linear one (because the policy is fixed). One may view this as an algorithm with two loops, the inner one performing the linear value iteration for a fixed policy and the outer one updating the policy at the end of it. In practice, of course, the update of the outer loop cannot be kept waiting forever while the inner loop algorithm converges asymptotically to the desired cost to go function. Various *ad hoc* schemes have been proposed to ensure reasonable behaviour (Barto *et al* 1995). Our aim here is to propose a variant of the actor–critic algorithm based on some recent results on stochastic approximation with two time scales (Borkar 1996). The idea here is to operate the inner and outer loops with different step-size schedules, so that the inner loop moves on a faster effective time scale than the outer loop. This ensures that while the inner loop sees the current policy in the outer loop as quasi-static, the outer loop sees the value iteration of the inner loop as essentially equilibrated. This provides an actor–critic scheme that is asymptotically exact, at least in principle.

Having said all this, we should hasten to add that the simulation-based algorithms are not without problems. The first major problem is that many of these simulations call for a parallel, distributed implementation of the algorithm. This throws up issues like asynchronism and interprocessor communication delays. It is well-known that even simple, innocuous iterations that perform ideally in a centralised implementation can go haywire in a parallel, distributed environment (Chazan & Miranker 1969). Fortunately, a considerable body of work is now available on conditions under which one may still get the desired convergence (Bertsekas & Tsitsiklis 1989; Borkar 1994). We apply these ideas to the present algorithm to underscore conditions under which the desired convergence is preserved in a parallel distributed set-up.

The second problem is the familiar 'curse of dimensionality' which looms much larger in simulation-based algorithms. This is because they involve iteration of vectors indexed by state and action, not state alone, which increases the dimensionality many-fold (not to mention that the algorithm perforce is restricted to finite state and action space case). Thus usually the algorithm must be accompanied by an approximation scheme to make it tractable. The traditional approach would be state aggregation whereby one clubs parts of state space into single meta-states. An alternative approach gaining currency is to approximate the value function directly (Schweitzer & Seidman 1985; Tsitsiklis & Van Roy 1996). This is appealing in view of the recently established function approximation properties of neural networks and good algorithms for neural network training that can be exploited here. Nevertheless, certain counterexamples in literature (Tsitsiklis & Van Roy 1996) suggest that the approach is not without its problems. We shall discuss these issues later in this paper.

The paper is organised as follows. The next section briefly reviews the MDP paradigm for discounted cost problem and describes our variant of actor–critic algorithm associated with it. Section 3 recalls some key results concerning stochastic approximation, from Borkar (1996) and Borkar (1994) respectively. These are used in § 4 for the convergence analysis of the algorithm and its asynchronous version. Section 5 describes an approximation scheme based on state aggregation. Section 6 presents a simulation example. Section 7 concludes with a brief discussion of further research issues.

This paper derives much from Borkar (1994; 1996) in terms of technique. Because of this, we have opted in favour of giving sketches of proofs instead of complete details. Giving the latter would require reproducing the aforementioned references in totality, a considerable overhead in terms of length and mathematical abstraction. Needless to say, we have pointed out the minor variations as and when needed. Complete mathematical details can be found in Rao (1996).

2. MDPs and the actor–critic algorithm

We begin by recapitulating some well-known facts about MDPs. Puterman (1994) is a good general reference for the material.

Let $S = \{1, 2, \ldots, s\}$, $A = \{a_0, a_1, \ldots, a_r\}$ be prescribed finite sets and $p : S \times S \times A \to [0, 1]$ a map satisfying

$$p(i, j, a) \in [0, 1], \quad \sum_k p(i, k, a) = 1 \quad \forall i, j, a.$$

An MDP (equivalently, a controlled Markov Chain) on state space S, with action space A and transition probability function $p(\cdot, \cdot, \cdot)$, is an S-valued random process $X_n, n \geq 0$, satisfying

$$P(X_{n+1} = j / X_k, Z_k, k \leq n) = p(X_n, j, Z_n) \quad \forall n \geq 0,$$

where $\{Z_n\}$ is an A-valued 'control' process. If $\{Z_n\}$ is of the form $Z_n = v(X_n), n \geq 0$, for some map $v : S \to A$, we call $\{Z_n\}$, or, by abuse of terminology, the map v itself, a stationary policy. More generally, if for each n, Z_n is conditionally independent of $X_m, Z_m, m < n$, given X_n, we call it a stationary randomised policy and identify

it with the map $\varphi : S \to \mathcal{P}(A)$ ($\mathcal{P}(\cdots)$ = the space of probability vectors on '$\cdots$') which gives the conditional law of Z_n given X_n. For $i \in S, a \in A$, let $\pi(i,a)$ denote the ath component of $\varphi(i)$. Under a stationary policy v (resp., a stationary randomised policy φ), $\{X_n\}$ is a time-homogeneous Markov chain with transition probabilities $[[p(i,j,v(i))]]$ (resp., $[[q(i,j,\varphi(i))]]$ where $q(i,j,\varphi(i)) = \sum_{a \in A} p(i,j,a)\pi(i,a)$). By a further abuse of terminology, we identify the stationary randomised policy φ with the vector $\pi = [\pi(i,a)]$, where the elements are ordered lexicographically. Note also that the class of stationary randomised policies contains the class of stationary policies, since the latter correspond to the case when each $\varphi(i)$ is concentrated at a single point in A.

We shall consider the infinite horizon discounted cost control problem. In this a discount factor $\alpha \in (0,1)$ and a running cost function $k : S \times A \to R$ are prescribed and the aim is to minimize over all admissible $\{Z_n\}$ the quantity

$$E\left[\sum_{n=0}^{\infty} \alpha^n k(X_n, Z_n)\right].$$

Define the 'value function' $V : S \to R$ by: For $i \in S$,

$$V(i) = \min E\left[\sum_{n=0}^{\infty} \alpha^n k(X_n, Z_n)/X_0 = i\right],$$

the minimum being over all admissible $\{Z_n\}$. It is known that V is the unique solution to the dynamic programming equations

$$V(i) = \min_a \left[k(i,a) + \alpha \sum_j p(i,j,a)V(j)\right], \quad i \in S.$$

Furthermore, any $v : S \to R$ satisfying: $v(i)$ attains the minimum in the rhs of the above, yields an optimal stationary policy, optimal for all initial conditions. In fact, $\{Z_n\}$ is optimal if and only if with probability one,

$$Z_n \in \arg\min(k(X_n, \cdot) + \alpha \sum_j p(X_n, j, \cdot)V(j)).$$

The key to the control problem therefore lies in finding $V(\cdot)$. Two standard approaches for this are value iteration and policy iteration. (A third approach to MDPs reduces them to linear programming problems. We do not consider this approach here.) The value iteration starts with an initial guess V_0 and iterates as per

$$V_{n+1}(i) = \min_a \left[k(i,a) + \alpha \sum_j p(i,j,a)V_n(j)\right], \quad i \in S,$$

for $n \geq 0$. Using Banach contraction mapping theorem, it is easy to show that $V_n \to V$ at an exponential rate.

The policy iteration scheme, on the other hand, starts with an initial guess $v : S \to A$ for an optimal policy and improves upon it iteratively as follows: At nth iteration

Step 1: Compute $V_n : S \to R$ defined by

$$V_n(i) = E\left[\sum_{m=0}^{\infty} \alpha^m k(X_m, Z_m)/X_0 = i\right], \quad i \in S,$$

the expectation being under the stationary policy $Z_m = v_n(X_m)$, $m \geq 0$. This is done by solving the linear equations

$$V_n(i) = k(i, v_n(i)) + \alpha \sum_j p(i, j, v_n(i)) V_n(j), \quad i \in S.$$

Step 2: Find $v_{n+1} : S \to R$ by

$$v_{n+1}(i) \in \arg\min \left[k(i, \cdot) + \alpha \sum_j p(i, j, \cdot) V_n(j)\right], \quad i \in S.$$

One can show that the cost strictly decreases at each iterate as long as V_n is suboptimal, ensuring convergence of $v_n(\cdot)$, $V_n(\cdot)$ to the optimal pair.

We now derive the appropriate 'simulation-based' version of this. There are some key differences between them which need to be underscored. The first, of course, is that we replace each summation involving $p(i, \cdot, a)$ by a simulated transition as per that probability vector. In order for this to work, the algorithm should do some averaging. This is ensured by using an incremental version which makes only a small change in current iterates at each step, weighted by a stochastic approximation – like decreasing step-size. Secondly, we operate with stationary randomised policies rather than stationary policies so that simple update equations for the probability vectors therein can be written. Finally, the linear system of step 1 is replaced by an iterative scheme for its solution before incorporating it into the simulation based scheme. This scheme is a 'stationary value iteration' given by

$$V_n^{m+1}(i) = k(i, v_n(i)) + \alpha \sum_j p(i, j, v_n^m) V_n^m(i), \quad i \in S,$$

for $m \geq 0$. This forms the 'inner loop' of the algorithm, wherein m is being updated for each fixed n. The equilibrium value $V_n(\cdot)$ to which $\{V_n^m(\cdot)\}$ will converge is then passed on to the outer loop for updating the policy. The crux of the algorithm we propose is to achieve this two-tier structure by using two different time scales.

The 'centralized' version of our variant of the actor–critic algorithm is as follows. Let $\{a(n)\}$, $\{b(n)\}$ be decreasing sequences in (0,1) satisfying

$$\sum_n a(n) = \sum_n b(n) = \infty \ , \sum_n a(n)^2, \sum_n b(n)^2 < \infty, \quad a(n) = o(b(n)).$$

Fix $a_0 \in A$ and let P denote the projection of an r-vector onto the simplex $D = \{[x_1, \ldots, x_r] \mid x_i \geq 0, \ \forall i, \ \sum_i x_i \leq 1\}$. Let $\hat{\pi}(i)$ denote the r-vector $[\pi(i, a_1), \ldots, \pi(i, a_r)]$. Let e_a denote the r-vector whose components are indexed by elements of $A \setminus \{a_0\}$, with its component indexed a as 1 and all other components equal to 0. The algorithm starts with an initial pair $V_0(\cdot) \in R^s$ and $\pi_0(i, a)$, $i \in S$, $a \in A \setminus \{a_0\}$, and iterates according to

$$V_{n+1}(i) = (1 - b(n)) V_n(i) + b(n)[\bar{k}(i, \varphi_n(i)) + \alpha V_n(\xi_n(i))],$$

$i \in S$, and for each $i \in S$, $a \in A \setminus \{a_0\}$,

$$\hat{\pi}_{n+1}(i) = P\left(\hat{\pi}_n(i) + a(n)\left(\sum_{a' \in A\setminus\{a_0\}} (V_n(i) - k(i,a') - \alpha V_n(\eta_n(i,a')))\pi_n(i,a')e_{a'} + \phi'(n)\right)\right)$$

$$\pi_{n+1}(i,a_0) = 1 - \sum_{a \neq a_0} \pi_{n+1}(i,a),$$

for $n \geq 0$, with $\varphi(i) \stackrel{\Delta}{=} \pi_n(i,\cdot) \in \mathcal{P}(A)$ and $\bar{k}(i,\varphi) = \sum_a \pi(i,a)k(i,a)$ for $\varphi \in \mathcal{P}(A)$. Furthermore, $\xi_n = [\xi_n(1), \ldots, \xi_n(s)]$, $\eta_n = [[\eta_n(i,a)]]$ are resp. S^s, $S^{s\times r}$-valued random variables conditionally independent of each other given ξ_i, η_i, $i < n$, with the corresponding conditional distributions equal to

$$\prod_{j=1}^{s} q(j,.,\varphi_n(j)) \text{ and } \prod_{j\in S}\prod_{a\in A} p(j,.,a)$$

respectively. The reader may verify that this is in confirmation with our verbal description earlier. A small modification, however, is warranted. If any $\pi_n(i,\cdot)$ is on a face of the simplex $\mathcal{P}(A)$, it will remain there thereafter. To avert this, a small diminishing noise $\phi'(n)$, with a Lebesgue continuous law, is added to push it away from the stable manifold of the unstable equilibrium points on the face of the simplex.

The distributed, asynchronous version is more complicated and needs additional notation and assumptions. To start with, let I_1, I_2 be sets of subsets of S, $S \times A \setminus \{a_0\}$ resp. that together cover resp. S, $S \times A \setminus \{a_0\}$. Let $\{Y_n\}$, $\{Z_n\}$ be resp. I_1-, I_2- valued processes with the interpretation: Y_n is the set of $i \in S$ such that $V_n(i)$ gets updated at time n and Z_n is the set of $(i,a) \in S \times A \setminus \{a_0\}$ such that $\pi_n(i,a)$ gets updated at time n.We impose on these processes the condition: There exists a deterministic $\Delta > 0$ such that with probability one,

$$\liminf_{n\to\infty} \frac{1}{n}\sum_{m=0}^{n-1} I\{i \in Y_m\} \geq \Delta, \quad i \in S,$$

$$\liminf_{n\to\infty} \frac{1}{n}\sum_{m=0}^{n-1} I\{(i,a) \in Z_m\} \geq \Delta, i \in S, a \in A.$$

This ensures that the distributed asynchronous version updates all components comparably often in a precise sense. (See Borkar 1994 for a graph theoretic sufficient condition for the above to hold.)

Secondly, we introduce delays $\tau_n(i,j)$, $\bar{\tau}_n(i,a,j)$, $\hat{\tau}(i,a)$ taking values in $[0, 1, \ldots, \min(n,N)]$, $N \geq 0$, with the assumption: $\tau_n(i,i) = 0$ $\forall i,n$. The idea is: Each component of the iteration is updated by a fixed processor, which receives the updates from other processors with random but bounded interprocessor communication delays. Thus the processor updating $V_n(i)$ receives at time n, $V_{n-\tau_n(i,j)}(j)$ and not $V_n(j)$. Similarly, the processor updating $\pi_n(i,a)$ receives at time n, $V_{n-\bar{\tau}_n(i,a,j)}(j)$ and not $V_n(j)$. Finally, the

processor updating $V_n(i)$ at time n receives $\pi_{n-\hat{\tau}_n(i,a)}(i,a)$ and not $\pi_n(i,a)$. Assume that $\{\tau_n(i,j), \bar{\tau}_n(i,a,j), \hat{\tau}_n(i,a)\}$ are independent of ξ_m, η_m $m < n$ for each n. The latter are defined as before, except that the conditional law of ξ_n given ξ_m, η_m, $m < n$ gets replaced by $\prod_{j\in S} q(j,\cdot,\varphi_n(j))$ where

$$\varphi_n(i) = \pi_{n-\hat{\tau}(i,\cdot)}(i,\cdot),\ i \in S.$$

(It should be remarked that the boundedness condition on delays can be replaced by a mild conditional moment bound as in Borkar 1994 at the expense of additional technicalities.)

Finally, introduce for $n \geq 0$,

$$\nu_1(i,n) = \sum_{m=0}^{n} I\{i \in Y_m\},\ i \in S,$$

$$\nu_2(i,a,n) = \sum_{m=0}^{n} I\{(i,a) \in Z_m\},\ i \in S,\ a \in A.$$

Note that the processor updating $V_n(i)$ (resp. $\pi_n(i,a)$) knows $\nu_1(i,n)$ (resp. $\nu_2(i,a,n)$) at time n (that being the number of updates he has performed till then), even when he does not know n, i.e., the 'global clock'.

The distributed, asynchronous version of the algorithm then is: for $i \in S$, $a \in A \setminus \{a_0\}$,

$$V_{n+1}(i) = V_n(i) + b(\nu_1(i,n))[\bar{k}(i,\varphi_n(i)) + \alpha V_{n-\tau_n(i,\xi_n(i))}(\xi_n(i)) - V_n(i)]I\{i \in Y_n\},$$

$$\hat{\pi}_{n+1}(i) = P\left(\hat{\pi}_n(i) + \sum_{a' \in A\setminus\{a_0\}} a(\nu_2(i,a,n))((V_n(i) - k(i,a) - \alpha V_{n-\bar{\tau}(i,a,\eta_n(i,a))}(\eta_n(i,a)))\pi_n(i,a) + \phi(n))I\{(i,a) \in Z_n\}e_{a'}\right),$$

$$\pi_{n+1}(i,a_0) = 1 - \sum_{a \neq a_0} \pi_{n+1}(i,a),$$

$n \geq 0$ and $\{\phi(n)\}$ is a random sequence converging to zero. The role of this sequence is same as that of $\phi(n)$ in synchronous algorithm. For this algorithm, we shall impose the following additional restrictions on $\{a(n)\}$, $\{b(n)\}$: Let $\{c(n)\}$ denote $\{a(n)\}$ or $\{b(n)\}$. Then

(1) There exists $r \in (0,1)$ such that $\sum_n c(n)^{1+r} < \infty$.

(2) For $x \in (0,1)$, $\sup_n c([xn])/c(n) < \infty$, where $[\cdots]$ stands for 'the integer part of $\cdots$'.

(3) For $x \in (0,1)$ and $A(n) = \sum_{m=0}^{n} c(m)$, $A([yn])/A(n) \to 1$ uniformly in $y \in [x,1]$.

Examples of $\{c(n)\}$ satisfying the above are: $1/n$, $1/n\ln(n)$, $\ln(n)/n$ etc., with modification for $n = 0, 1$ where needed.

We shall analyse these algorithms in § 4 after a brief review of some relevant topics in stochastic approximation in the next section.

3. Stochastic approximation

This section briefly recalls some recent results in stochastic approximation algorithms needed for our work. The stochastic approximation algorithm in its simplest form is the d-dimensional iteration

$$X(n+1) = X(n) + a(n)(h(X(n)) + M(n)), \ n \geq 0, \tag{1}$$

where $\{a(n)\}$ is as before and $\{M(n)\}$ is a sequence of integrable random variables satisfying

$$E[M(n)/X(m), \ m \leq n, \ M(m), \ m < n] = 0, \ n \geq 0. \tag{2}$$

The convergence of this algorithm to a desired limit is usually established by first establishing separately that with probability one,

$$\sup_n \mid X(n) \mid < \infty, \ \sum_n a(n)M(n) < \infty. \tag{3}$$

Given these, one way to analyse its asymptotic behaviour is by showing that it asymptotically tracks the ordinary differential equation (ODE) given by

$$\dot{x}(t) = h(x(t)).$$

Assume h to be Lipschitz with linear growth, ensuring that this ODE has a unique solution for any initial condition, defined for all $t \geq 0$.Suppose this ODE has a globally asymptotically stable attractor J. Then by converse Liapunov theorem (see, e.g., Yoshizawa 1966) there exists a continuously differentiable $V : R^d \to R^+$ satisfying $V(x) \to \infty$ as $\|x\| \to \infty$ and $\nabla V(x) \cdot h(x) < 0$ for $x \notin J$. Now, given $T, \delta > 0$, call a bounded measurable function $y(\cdot) : R^+ \to R^d$ a (T, δ)-perturbation of this ODE if there exist $0 = T_0 < T_1 < T_2 < \cdots T_i \uparrow \infty$, such that $T_{j+1} \geq T_j + T$ and there exist solutions $x^j(\cdot)$ of the ODE on each interval $[T_j, T_{j+1}]$ such that

$$\sup_{t \in [T_j, T_{j+1}]} \|x^j(t) - y(t)\| < \delta.$$

Then one has:

Lemma 1 (Hirsch 1989). *For every $\epsilon > 0$, $T > 0$, there exists a $\delta_0 > 0$ such that for any $\delta \in (0, \delta_0)$, every (T,δ)-perturbation of the ODE converges to the ϵ-neighbourhood of J.*

The idea of the proof is: The Liapunov function V must decrease by a minimum positive quantity along every $x^j(\cdot)$ that does not intersect the ϵ-neighbourhood of J and therefore along the corresponding patch of $y(\cdot)$. This can happen for at most finitely many consecutive j's, so $y(\cdot)$ must eventually intersect this neighbourhood. Analogous considerations show that it cannot move away too much once it has done so. (See the appendix of Borkar 1996 for details.)

The convergence analysis of the stochastic approximation algorithm now hinges on the following 'time-scaling' argument: Let $t(0) = 0$, $t(n) = \sum_{m=0}^{n-1} a(m)$, and pick $m(n)$ according to: $m(0) = 0$, $m(n) = \min\{t(k) \mid t(k) \geq t(m(n)) + T\}$. Set $T_j = t(m(j))$,

$j \geq 0$. Define $y(t)$, $t \geq 0$, by: $y(t(m)) = X(m)$, $m \geq 0$, with linear interpolation on $[t(m), t(m+1)]$, $m \geq 0$. Let $x^j(\cdot)$ be the solution of the ODE on $[T_j, T_{j+1}]$ with $x^j(T_j) = y(T_j)$, $j \geq 0$. Then $y(\cdot)$ on $[T_j, T_{j+1}]$ may be viewed as an Euler approximation of the ODE with a nonuniform but decreasing (with j) step-size, modulo an error term due to $\{M(n)\}$ that also becomes asymptotically negligible thanks to (3). The above lemma then applies 'eventually' (i.e., for sufficiently large j) for each $\epsilon \geq 0$, ensuring $X(n) \to J$ with probability one.

Now consider a 'two time-scale' variant of the basic algorithm:

$$X(n+1) = X(n) + a(n)(F(X(n), Y(n)) + M(n+1)),$$
$$Y(n+1) = Y(n) + b(n)(G(X(n), Y(n)) + M'(n+1)),$$

where F, G are Lipschitz with linear growth and $M(n)$, $M'(n)$ are integrable random variables uncorrelated with the past (i.e., satisfying (2)) and $a(n) = o(b(n))$. Suppose with probability one, the following hold:

$$\sup_n | X(n) | < \infty, \quad \sum_n a(n)M(n) < \infty,$$
$$\sup_n | Y(n) | < \infty, \quad \sum_n a(n)M'(n) < \infty.$$

Also suppose that for each x, the ODE,

$$\dot{y}(t) = G(x, y(t)), \tag{4}$$

has a unique globally asymptotically stable equilibrium point $\lambda(x)$ where $\lambda(\cdot)$ is Lipschitz continuous and the ODE,

$$\dot{x}(t) = F(x(t), \lambda(x(t))), \tag{5}$$

has a unique globally asymptotically stable attractor J.

Theorem 1. *With probability one,* $(X(n), Y(n)) \to \{(x, \lambda(x)) \mid x \in J\}$.

The proof can be found in Borkar (1996) for the case when J is a singleton and extends easily to the more general case. The idea is to mimic the above time scaling argument first with $b(n)$ (i.e., $t(n) = \sum_{i=0}^{n} b(i)$), so that the interpolated trajectories track the ODE

$$\dot{x}(t) = 0, \quad \dot{y}(t) = G(x(t), y(t)),$$

and then again with $a(n)$ (i.e., $t(n) = \sum_{i=0}^{n} a(i)$), so that the interpolated $\{X(n)\}$ tracks (5). Lemma 1 is used in each case in the obvious manner. Thus the fast component $\{Y(n)\}$ sees the slow component $\{X(n)\}$ as quasi-static, while the slow component sees the fast component as 'essentially equilibrated'.

The distributed, asynchronous version of the algorithm (1) is as follows: If $X_i(n)$ is the ith component of the vector $X(n)$, it is updated as per

$$X_i(n+1) = X_i(n) + a(\nu(i,n))(h(X_1(n - \tau_n(i,1)), \ldots, X_d(n - \tau_n(i,d))) + M(n))I\{i \in Y_n\}$$

where $\{Y_n\}$, $\{\tau_n(i,j)\}$, $\{M(n)\}$ are as before.

For this case, the rather intricate analysis of Borkar (1994) shows that a suitably interpolated version of $\{X(n)\}$, if bounded, tracks the ODE

$$\dot{x}(t) = \frac{1}{d}h(x(t)).$$

The scalar $1/d$ up front amounts to linear time scaling that does not alter the qualitative behaviour. Thus $X(n) \to J$ with probability one as before. (It should be remarked that the algorithm considered in Borkar 1994 is slightly more restrictive than (1) or (5), but the same arguments go through nevertheless.)

4. Convergence analysis

In this section we shall adapt the ideas of the preceding section for the convergence analysis of the actor–critic algorithm proposed in § 2. As already mentioned earlier, only a sketch of the proofs will be given.

Define the map $F : \mathcal{P}(A)^s \times R^s \to R^s$ by $F(\cdot,\cdot) = [F_1(\cdot,\cdot), \ldots, F_s(\cdot,\cdot)]^T$ where $F_i(z,x) = \sum_a z(i,a)k(i,a) + \alpha \sum_j \sum_a z(i,a)p(i,j,a)x_j$ for $x = [x_1, \ldots, x_s]^T$ and $z = [[z(i,a)]]$, $i \in S, a \in A$ with $z(i,\cdot) \in \mathcal{P}(A)$ $\forall i$. Also define the following norms on R^s

$$\|x\|_p = \left(\frac{1}{s}\sum_{i=1}^{s} |\, x \,|^p\right)^{\frac{1}{p}}, \quad p \in (1,\infty),$$

$$\|x\|_\infty = \max_i |\, x_i \,| .$$

Then it is easily verified that for each fixed z, the following contraction condition holds:

$$\|F(z,x) - F(z,y)\|_\infty \leq \alpha \|x - y\|_\infty , \; x, y \in R^s.$$

The first iteration of the centralised algorithm can now be written as

$$V_{n+1} = V_n + b(n)(F(\pi_n, V_n) - V_n) + b(n)M(n), \quad n \geq 0 \tag{6}$$

for suitably defined $M(n)$ which will satisfy (2) (with $X(m) \stackrel{\Delta}{=} V_m$).

Lemma 2. With probability one, the iterates of the centralized or asynchronous algorithm remain bounded.

Proof. We consider the asynchronous case, as the centralized case is a special case thereof. The iterates of $\{\pi_n\}$ are bounded anyway because of the projection P onto a bounded set. That V_n remain bounded with probability one follows exactly as in theorem 1, pp. 190–191, of Tsitsiklis (1994). (In Tsitsiklis 1994, the algorithm slightly differs from (6) insofar as $F(\pi_n, V_n)$ would be replaced by $\bar{F}(V_n)$ for some $\bar{F}$ satisfying (say) $\|\bar{F}(x) - \bar{F}(y)\|_\infty \leq \alpha\|x - y\|_\infty$. Nevertheless, exactly the same proof applies.) □

In turn, this implies that $\{M(n)\}$, which is defined in terms of V_n, remains bounded with probability one. A direct calculation shows that in addition

$$\psi(n) \stackrel{\Delta}{=} E[M(n)^2 / V_m, \pi_m, \tau_m(i,j), \bar{\tau}_m(i,a,j), \hat{\tau}_m(i,a), Y_m, Z_m,$$
$$m \leq n, i, j \in S, a \in A]$$

remain bounded with probability one. Since $\sum_n b(n)^2 < \infty$, we then have $\sum_n b(n)^2\psi(n)^2 < \infty$ with probability one. The property (2) renders the process

$$\sum_{i=0}^{n} b(i)M(i), n \geq 0,$$

a martingale and the foregoing then ensures that its quadratic variation process (see Neveu 1975, for a definition) is convergent with probability one. Proposition VII-3-(c), pp. 149–150 of Neveu (1975) then ensures that

$$\sum_{i=0}^{\infty} b(i)M(i)$$

is convergent, and hence finite with probability one. This, in conjunction with lemma 2 above, completes our verification of (3) for iteration (6).

These considerations will enable us to use the results of the preceding section. Prior to doing so, we look at the expected ODE limits. The first one to be considered is the counterpart of (4) in our set-up, which is

$$\dot{x}(t) = F(\pi, x(t)) - x(t) \tag{7}$$

for F as above and a fixed $\pi \in \mathcal{P}(A)^s$.

Lemma 3. Equation (7) *has a unique globally asymptotically stable equilibrium point* $\bar{V}_\pi$ *given by*

$$\bar{V}_\pi(i) = E\left[\sum_{m=0}^{\infty} \alpha^m k(X_m, Z_m)/X_0 = i\right],$$

the expectation being with respect to the stationary randomised policy π.

Proof. Standard contraction mapping arguments show that $F(\pi, \cdot)$ has a unique fixed point and usual dynamic programming type arguments show that $\bar{V}_\pi$ defined above must be it. Now for $p \in (1, \infty)$, explicit differentiation leads to

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\|x(t) - \bar{V}_\pi\|_p &= \|x(t) - \bar{V}_\pi\|_p^{1-p}(-\|x(t) - \bar{V}_\pi\|_p \\ &\quad + \frac{1}{s}\sum_{i=1}^{s}(x_i(t) - \bar{V}_\pi(i))^{p-1}(F_i(\pi, x(t)) - F_i(\pi, \bar{V}_\pi)) \\ &\leq -\|x(t) - \bar{V}_\pi\|_p + \|F(\pi, x(t)) - F(\pi, \bar{V}_\pi)\|_p\end{aligned}$$

where the inequality follows from an application of Hölder's inequality. Let $p \to \infty$. Using continuity of $p \to \|x\|_p$ on $[1, \infty]$ and the contraction property of $F(\pi, \cdot)$ under the $\|\cdot\|_\infty$-norm, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\|x(t) - \bar{V}_\pi\|_\infty \leq -(1-\alpha)\|x(t) - \bar{V}_\pi\|_\infty,$$

which implies the claim by standard arguments. □

Note that since $\bar{V}_\pi$ is characterized by the linear system of equations $F(\pi, \bar{V}_\pi) = \bar{V}_\pi$, it depends smoothly on the coefficients thereof, hence on π. In particular, the map is Lipschitz on $\mathcal{P}(A)^S$.

Define a vector field $G(\cdot)$ on $\mathcal{P}(A)^S$ by:$G(\cdot) = [[G_{ia}(\cdot)]]$ for $i \in S$, $a \in A$, where

$$G_{ia}(\pi) = \pi(i,a)\left(V_\pi(i) - k(i,a) - \alpha \sum_j p(i,j,a)V_\pi(j)\right).$$

The second ODE (the counterpart of (5)) that we consider is

$$\dot{\pi}(t) = G(\pi(t)). \tag{8}$$

It is easy to verify that this ODE will always remain in $\mathcal{P}(A)^S$ if $\pi(0)$ is. Furthermore, it will converge to equilibrium points, the latter being the set of those $\pi \in \mathcal{P}(A)^S$ that satisfy: for each (i,a), either $\pi(i,a) = 0$ or $k(i,a) + \alpha \sum_j p(i,j,a)V_\pi(j) = V_\pi(i)$. A little thought shows that the stable equilibrium points are those for which, for each (i,a), either $\pi(i,a) = 0$ or

$$V_\pi(i) = \min_a \left(k(i,a) + \alpha \sum_j p(i,j,a)V_\pi(j)\right).$$

Since not all $\pi(i,a)$ can be zero for a given i, the above equality holds for all i.Hence the only stable equilibrium points are the π for which V_π equals V. These are precisely the optimal stationary randomised policies.

Consider the centralized algorithm.

Theorem 2. *With probability one, the centralized algorithm converges to* $\{(V,\pi) \mid \pi$ *is an optimal stationary randomised policy*$\}$.

Proof. We can adopt the 'two-time' scale results of the preceding section with one difference. The difference here is the presence of the projection operator P in the algorithm. This, however, is a standard feature of many stochastic approximation algorithms and the theory thereof is well-understood; see, e.g., Kushner & Clark (1978). As in the theorem 5.3.1, p. 191, of Kushner & Clark, we can conclude that the limiting ODE tracked by a suitably interpolated version of $\{\pi_n\}$ (and hence by $\{\pi_n\}$ itself) is given by the $R^{s\times(r-1)}$-dimensional ODE

$$\dot{\pi}(t) = \bar{P}(\bar{G}(\pi(t))),$$

where $\bar{G}_{ia}(\pi) = G_{ia}([\pi[1,\cdot]\vdots 1 - \sum_{b\neq a_0}\pi(1,b)\vdots\pi(2,a)\vdots 1 - \sum_{b\neq a_0}\pi(2,b)\vdots\cdots\vdots\pi(s,a)\vdots 1 - \sum_{b\neq a_0}\pi(s,b)])$, $j \in S$, and $\bar{P}(h(\cdot))$, for a vector field $h(\cdot)$ on $\mathcal{P}(A)^S$ is defined by

$$\bar{P}(h(y)) = \lim_{0<\delta\to 0} \frac{P(y + h(y)\delta) - P(y)}{\delta}.$$

If the limit is non-unique, one considers the set of all limit points and treats the above as a differential inclusion rather than an ODE. Fortunately for us, $\bar{P}(\bar{G}(\pi))$ is well-defined and in fact equals $G(\pi)$, as can be easily verified. Thus the limiting ODE is

$$\dot{\pi} = G(\pi(t)).$$

We now augment $\pi(t)$ by adjoining s additional components $\pi(i, a_0) = 1 - \sum_a \pi(i, a)$, $i \in S$. Denote the enlarged vector by $\pi(t)$ again by abuse of notation. It is easy to verify that thus redefined, $\pi(\cdot)$ satisfies (8). The convergence for the two time scale stochastic algorithms can now be invoked to claim that with probability one, $\{\pi_n\}$ converges to the set of equilibrium points of (8). The final step is to show that it will in fact converge to the set of stable equilibrium points of (8) with probability one (Pemantle (1990), see also Brandière & Duflo (1996)). To show this, all we need is that the set of points that get attracted to an unstable equilibrium point have zero probability. It is easy to see that Lebesgue measure of the stable manifold of an unstable equilibrium point is zero. Thus it is enough if the noise $\phi(n)$ has positive density with respect to Lebesgue measure in a shrinking neighbourhood of the origin. The claim follows. □

Theorem 3. *The claims of theorem* 2 *also hold for the decentralized algorithm under additional hypotheses stipulated in* § 3.

This follows simply by combining the foregoing with the scheme of Borkar (1994). This completes our convergence analysis of the algorithm.

5. Approximation issues

As mentioned in the introduction, in many applications the 'curse of dimensionality' forces one to interface the above and other simulation-based algorithms with an approximation scheme. Here we sketch a scheme based on state aggregation, adapted from Tsitsiklis & Van Roy (1994) where it is proposed as a 'look-up table scheme' in the context of Q-learning.

The idea is to partition the state space S into disjoint nonempty subsets $S_1, S_2, \ldots, S_m$, each identified as an aggregated state. For each j, let $\beta_j(\cdot)$ be a probability vector on S_j with $\beta_j(i) > 0$, $\forall i \in S_j$. The centralized version of the algorithm runs as follows: At each n, generate for each j, $1 \leq j \leq m$, random variables $(X_j^n, \xi_j^n, \eta_{ja}^n, a \in A)$ with law

$$P(X_j^n = i_j, \xi_j^n = k_j, \eta_{ja}^n = x_{aj}, a \in A, j = 1, 2, \ldots, m) = \prod_{j=1}^{m} \beta_j(i_j) q(i_j, k_j, \varphi_n(j)) \prod_{a \in A} p(i_j, x_{aj}, a)$$

independently of the past, where $\varphi_n(j) = \pi_n(j, .)$. Define $\bar{\eta}_{ja}^n \in \{1, 2, \ldots, m\}$ by $\bar{\eta}_{ja} = k$ if $\eta_{ja}^n \in S_k$. The iteration at time n is: for each j, $1 \leq j \leq m$,

$$V_{n+1}(j) = (1 - b(n)) V_n(j) + b(n)(\bar{k}(X_j^n, \varphi_n(j)) + \alpha V_n(\xi_j^n)),$$

$$\hat{\pi}_{n+1}(j) = P\Bigg(\hat{\pi}_n(j) + a(n)\Bigg(\sum_{a \in A \setminus \{a_0\}} V_n(j) - \bar{k}(X_j^n, a) - \alpha V_n(\bar{\eta}_{ja}))\pi_n(j, a)e_0 + \phi(n)\Bigg)\Bigg),$$

$$\pi_{n+1}(j, a_0) = 1 - \sum_{a \neq a_0} \pi_{n+1}(j, a),$$

where P is the projection onto the appropriate r-dimensional simplex.

An analysis similar to the foregoing shows that with probability one, these iterates converge to $(V^*(\cdot), \pi^*(\cdot, \cdot))$, where for $j \in \{1, 2, \ldots, m\}$,

$$V^*(j) = \min_a \left[\sum_{i \in S_j} \beta_j(i)[k(i, a) + \alpha \sum_{l=1}^{m} V(l) \sum_{x \in S_l} p(i, l, a)] \right]$$

and support $(\pi^*(j, .)) \subset \arg\min$ of the rhs above. For error estimates for such schemes, see Tsitsiklis & Van Roy (1994)

A distributed version of this approximate scheme along the lines of the foregoing can also be analysed accordingly.

6. Simulation experiments

To demonstrate the convergence of the proposed algorithm we took two examples. The MDPs of these two examples are similar to those used to model admission control into an $M/M/1$ queue and routing to parallel $M/M/1$ queues respectively (Walrand 1988). In both these examples, in any iteration, only one component of V and one component of π are updated. The probability of update of a component in an iteration is equal for all components and is independent of everything else. Since $\pi_n(i, a)$ cannot change once it becomes zero, a small noise is added whenever it is sufficiently close to zero. The deterministic value iteration was used to find out optimal value function and policy. The algorithm was started with value function being identical to zero, and with all the randomised policy probabilities equal. The examples are described below.The notation used here is the one introduced in § 2. Interprocessor communication is assumed to be instantaneous (i.e., no delays).

6.1 *Example 1*

$$S = \{0, 1, \ldots, N\},$$
$$A = \{0, 1\},$$
$$p(i, j, a) = \begin{cases} 0, & \text{if } j < \max(j-1, 0) \text{ or } j > i+1, \\ \lambda, & \text{if } j = \min(i+1, N) \text{ and } a = 0, \\ & \text{or } j = i \text{ and } a = 1, \\ 1-\lambda, & \text{if } j = \max(i-1, 0), \end{cases}$$
$$k(i, a) = \begin{cases} i, & \text{if } a = 0, \\ i + c, & \text{if } a = 1. \end{cases}$$

For the purpose of simulation we took:

$$N = 10, \quad a(n) = (\{\lfloor \tfrac{n}{100} \rfloor + 1\}/n)^{\frac{2}{3}}$$
$$\lambda = 0.65,$$
$$c = 10, \quad b(n) = \{\lfloor \tfrac{n}{100} \rfloor + 1\}/n,$$
$$\alpha = 0.99.$$

The optimal value function and policy were:

$$V = [669.0261 \quad 679.4227 \quad 694.0252 \quad 711.0231 \quad 730.3492 \quad 751.9383 \quad 775.7268 \quad 801.6532 \quad 820.9869 \quad 831.7235 \quad 836.4438],$$

$$\pi(\cdot, 0) = [1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1],$$

$$\pi(i, 1) = 1 - \pi(i, 0) \quad i \in S.$$

After 2×10^6 iterations the result of our algorithm was

$$\hat{V} = [663.2563 \quad 673.3967 \quad 687.6990 \quad 708.0571 \quad 725.9087 \quad 748.0945 \quad 775.7268 \quad 796.7382 \quad 820.2483 \quad 830.5291 \quad 835.0053],$$

$$\hat{\pi}(\cdot, 0) = [1 \quad 0.9986 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.9871 \quad 1 \quad 0.9955],$$

$$\hat{\pi}(i, 1) = 1 - \hat{\pi}(i, 0) i \in S.$$

6.2 *Example 2*

$$S = \{(i, j),\ 0 \le i \le N,\ 0 \le j \le N\},$$

$$A = \{0, 1\},$$

$$p((i, j), (k, l), a) = \begin{cases} \lambda, & \text{if } k = \min(i + 1, N) \text{ and } j = l \\ & \quad \text{and } a = 0 \\ & \text{or } k = i \text{ and } l = \min(j + 1, N) \\ & \quad \text{and } a = 1, \\ \mu_1, & \text{if } k = \max(i - 1, 0) \text{ and } l = j, \\ 1 - \lambda - \mu_1, & \text{if } k = i \text{ and } l = \max(j - 1, 0), \\ 0, & \text{otherwise,} \end{cases}$$

$$k((i, j), a) = c_1 i + c_2 j.$$

For the purpose of simulation we took:

$$\begin{aligned} N &= 10, & a(n) &= (\{\lfloor \tfrac{n}{100} \rfloor + 1\}/n)^{\frac{2}{3}} \\ \lambda &= 0.65, & b(n) &= 0.1(\{\lfloor \tfrac{n}{100} \rfloor + 1\}/n) \\ c_1 &= 10, & c_2 &= 15, \\ \alpha &= 0.99, & \lambda &= 0.5, \\ \mu_1 &= 0.3, & \mu_2 &= 0.2. \end{aligned}$$

The optimal value function and policy were:

$$[[V(i, j)]] = 10^3 \times \begin{bmatrix} 3.3602 & 3.4448 & 3.5978 & 3.8137 & 4.0899 & 4.2539 \\ 3.4281 & 3.5179 & 3.6710 & 3.8863 & 4.1618 & 4.3018 \\ 3.5277 & 3.6183 & 3.7696 & 3.9833 & 4.2578 & 4.3875 \\ 3.6380 & 3.7171 & 3.8643 & 4.0761 & 4.3496 & 4.4937 \\ 3.7172 & 3.7922 & 3.9370 & 4.1474 & 4.4201 & 4.6036 \\ 3.7590 & 3.8328 & 3.9766 & 4.1864 & 4.4588 & 4.7032 \end{bmatrix},$$

$$[[\pi((i,j),0)]] = \begin{bmatrix} 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 0 \\ 0 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 0 \\ 0 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 0 \\ 0 \text{ to } 1 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 0 \\ 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 0 \\ 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 0 \text{ to } 1 \end{bmatrix}.$$

After 10^7 iterations the result of our algorithm was

$$[[\hat{V}(i,j)]] = 10^3 \times \begin{bmatrix} 3.3715 & 3.4610 & 3.6047 & 3.8221 & 4.1100 & 4.2684 \\ 3.4399 & 3.5317 & 3.6813 & 3.9045 & 4.1760 & 4.3166 \\ 3.5316 & 3.6327 & 3.7757 & 3.9971 & 4.2682 & 4.3978 \\ 3.6365 & 3.7207 & 3.8772 & 4.0869 & 4.3490 & 4.4987 \\ 3.7219 & 3.8000 & 3.9480 & 4.1499 & 4.4253 & 4.6193 \\ 3.7650 & 3.8431 & 3.9849 & 4.1919 & 4.4675 & 4.7109 \end{bmatrix},$$

$$[[\hat{\pi}((i,j),0)]] = \begin{bmatrix} 1.0000 & 0.9861 & 1.0000 & 1.0000 & 0.9900 & 0.0000 \\ 0.0000 & 1.0000 & 0.9900 & 1.0000 & 0.9844 & 0.0000 \\ 0.0000 & 0.9997 & 0.9910 & 0.9977 & 0.9994 & 0.0000 \\ 0.0000 & 1.0000 & 1.0000 & 1.0000 & 0.9977 & 0.0000 \\ 0.9975 & 0.9998 & 0.9884 & 0.9999 & 0.9999 & 0.0000 \\ 1.0000 & 1.0000 & 1.0000 & 0.9998 & 1.0000 & 0.0000 \end{bmatrix}.$$

Note that the convergence is slow, the price we pay for not using prior information about the transition mechanism as well as for asynchronism.

7. Further directions

We conclude by listing some promising research directions.

Average cost control: In many problems such as control of communication networks where quasi-equilibrium behaviour is desired, long-run average cost is preferred to discounted cost. It would be useful to extend the algorithm to the average cost setup, possibly using the techniques developed in Abounadi *et al* (1996a,b) for Q-learning.

Approximation based on compact representations: In addition to the approximation scheme proposed in § 5 above, Tsitsiklis & Van Roy (1996) also consider another scheme based on compact representations in the context of Q-learning. The idea is to directly approximate $V(\cdot)$ by a function belonging to a prescribed parameterized family and update the parameter in question rather than updating $V(\cdot)$ directly. In the actor–critic scheme, however, there is an additional iteration for $\pi_n(\cdot,\cdot)$. One can conceivably write a function approximation scheme for $\pi(\cdot,\cdot)$ using a parameterized family (such as a neural network – see, e.g., Santharam & Sastry 1997) and update the probabilities recursively. These possibilities need to be explored.

Feedback implementations: The algorithm we consider above is 'off-line', i.e., is based on a simulation run rather than an actual system being controlled in real time. One can convert it into an on-line (or feedback) adaptive control algorithm for a controlled Markov chain $X_n, n \geq 0$, by setting $Y_n = \{X_n\}$ $\forall n$ and letting $\pi_n(X_n, \cdot)$ be the actual randomised control law being implemented. A natural question then is whether the scheme is asymptotically optimal. (For the appropriate concept of 'asymptotically optimal' in the discounted framework, see Schäl 1987). Recall that the convergence of the algorithm to desired limits requires that all state-action pairs be tried sufficiently often. For states, this may happen automatically if suitable irreducibility conditions are met, even in the feedback case. But if the $\pi_n(\cdot, \cdot)$ converge rapidly (to the desired limit or otherwise) all the state-action pairs may not get updated frequently enough. A simple way out of this conflict is to modify the feedback law to a convex combination of $\varphi_n(\cdot)$ and the uniform distribution on A so as to ensure a minimum probability $\epsilon > 0$ of each $a \in A$ being picked. For $\epsilon > 0$ sufficiently small, the scheme will be nearly optimal within a prescribed tolerance. However, too small an ϵ may slow down convergence. Thus there is a trade-off involved. A potentially promising scheme is to start with a large $\epsilon \in (0, 1/r]$ (to ensure all state-action pairs being tried frequently) and then reduce it 'slowly' enough to ensure optimality. (Recall the simulated annealing algorithm for global optimization.) It is, however, a nontrivial task to capture the optimal rate of decrease of ϵ in a precise manner. These issues need further study. One should also add that presence of interprocessor communication delays causes nontrivial complications in the feedback case.

Rate of convergence: We have not provided any theoretical analysis of convergence rate. Since a stochastic approximation algorithm eventually tracks the associated ODE in a precise sense outlined in § 3, the convergence of its interpolated version to a given neighbourhood of the asymptotically stable limit of the ODE (assuming one exists) will closely mimic that of the ODE itself. The rate of the latter could be gauged from the Liapunov function approach. One must, however, invert the time-scaling $n \to t(n)$ to get the convergence behaviour of the original algorithm.

Even this may be worthless if 'eventually' is in too distant a future. There are other problems too: The ODE captures the averaging effect of the algorithm akin to the law of large numbers. But there can be fluctuations around the average behaviour of the 'central limit theorem' variety. For a two time scale algorithm, the time scales should be separated enough so that the slow one does not get swamped by the fluctuations of the fast one.

A related issue is the generally high variance of the stochastic approximation algorithms. An additional averaging can reduce this, see, e.g., Polyak (1990). This and other issues pertaining to improving the performance of the algorithm need careful study. To cut a long story short, the newly opened field of simulation-based algorithms for control offers many challenges both in theory and practice.

The work of the first author was supported by a Homi Bhabha Fellowship.

References

Abounadi J, Bertsekas D, Borkar V 1996a ODE analysis of stochastic algorithms involving sup-norm non-expansive maps (preprint)

Abounadi J, Bertsekas D, Borkar V 1996b Q-learning algorithms for average cost problems (preprint)

Barto A, Sutton R, Anderson C 1983 Neuron-like elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man Cybern.* 13: 835–846

Barto A, Bradtke S, Singh S 1995 Learning to act using real-time dynamic programming. *Artif. Intell. (Special Issue on Computational Theories of Interaction and Agency)* 72: 81–138

Benveniste A, Metivier M, Priouret P 1990 *Adaptive algorithms and stochastic approximations* (Berlin-Heidelberg: Springer-Verlag)

Bertsekas D, Tsitsiklis J 1989 *Parallel and distributed computation: Numerical methods* (Englewood Cliffs, NJ: Prentice Hall)

Borkar V 1994 *Asynchronous stochastic approximation. SIAM J. Control Optimization* (to appear)

Borkar V 1996 *Stochastic approximation with two time scales. Syst. Control Lett.* 29: 291–294

Brandière O, Duflo M 1996 Les algorithmes stochastiques contournent-ils les pièges? *Ann. Inst. Henri Poincarè* 32: 395–427

Chazan D, Miranker W 1969 Chaotic oscillations *Linear Algebra Appl.* 2: 199–222

Hirsch M 1989 Convergent activation dynamics in continuous time networks. *Neural Networks* 2: 331–349

Keerthi S S, Ravindran B 1994 A tutorial survey of reinforcement learning. *Sādhanā* 19: 851–889

Konda V 1996 *Learning algorithms for Markov decision processes.* Master's thesis, Dept. of Computer Science and Automation, Indian Institute of Science, Bangalore

Kushner H, Clark D 1978 *Stochastic approximation for constrained and unconstrained systems* (New York: Springer-Verlag)

Neveu J 1975 *Discrete parameter martingales* (Amsterdam: North Holland)

Pemantle R 1990 Non-convergence to unstable points in urn models and stochastic approximations. *Ann. Probab.* 18: 698–712

Polyak B 1990 New method of stochastic approximation type. *Autom. Remote Control* 51: 937–946

Puterman M 1994 *Markov decision processes* (New York: John Wiley)

Santharam G, Sastry P S 1997 A reinforcement learning neural network for adaptive control of Markov chains. *IEEE Trans. Syst. Man Cybern.* 27: 588–600

Schäl M 1987 Estimation and control of discounted dynamic programming. *Stochastics* 20: 51–71

Schweitzer P, Seidman A 1985 Generalized polynomial approximations in Markovian decision processes. *J. Math. Anal. Appl.* 110: 568–582

Tsitsiklis J 1994 Asynchronous stochastic approximation and Q-learning. *Mach. Learning* 16: 185–202

Tsitsiklis J, Van Roy B 1996 Feature-based methods for large scale dynamic programming. *Mach. Learning* 22: 59–94

Walrand J 1988 *Introduction to queueing networks* (Englewood Cliffs, NJ: Prentice Hall)

Watkins C 1989 *Learning from delayed rewards.* Ph D thesis, Cambridge University, Cambridge, England

Watkins C, Dayan P 1992 Q-learning. *Mach. Learning* 8: 279–292

Williams R, Baird L III 1990 A mathematical analysis of actor–critic architectures for learning optimal controls through incremental dynamic programming. In *Proc. Sixth Yale Workshop on Adaptive and Learning Systems*, New Haven, CT, pp 96–101

Yoshizawa T 1966 *Stability theory by Liapunov's second method* (Tokyo: Mathematical Society of Japan)