Available online at www.sciencedirect.com
ScienceDirect
Automatica 44 (2008) 1111–1119
automatica
www.elsevier.com/locate/automatica

Brief paper

New algorithms of the Q-learning type☆

Shalabh Bhatnagar[a,*], K. Mohan Babu[b]

[a]Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560 012, India
[b]Motorola India Electronics Ltd., Bangalore, India

Received 4 November 2006; received in revised form 16 May 2007; accepted 3 September 2007
Available online 21 December 2007

Abstract

We propose two algorithms for Q-learning that use the two-timescale stochastic approximation methodology. The first of these updates Q-values of all feasible state–action pairs at each instant while the second updates Q-values of states with actions chosen according to the 'current' randomized policy updates. A proof of convergence of the algorithms is shown. Finally, numerical experiments using the proposed algorithms on an application of routing in communication networks are presented on a few different settings.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Q-learning; Reinforcement learning; Markov decision processes; Two-timescale stochastic approximation; SPSA

1. Introduction

Markov decision process (MDP) is a general paradigm for solving dynamic decision making problems under uncertainty. Classical solution approaches such as policy iteration and value iteration for solving the associated Bellman equation for optimality require complete knowledge of the system model. Moreover, the computational requirements for solving the Bellman equation become prohibitive in the presence of large state and action spaces. Motivated by these considerations, there has been significant research in recent times on simulation based methods for solving the Bellman equation, that largely go under the name reinforcement learning or neuro-dynamic programming (Bertsekas & Tsitsiklis, 1996). The main idea here is to simulate transitions instead of computing transition probabilities that may be hard to obtain and to use parametric representations of the cost-to-go function. Alternatively, if one has access to real system data, the same may also be used directly in the associated algorithms. In Watkins and Dayan (1992), a simulation-based variant of the value iteration algorithm that goes under the name Q-learning was proposed. Here one estimates the state–action value function, also called the Q-value function, via simulation. In the infinite horizon discounted cost setting as we consider, the Q-value of a state–action pair corresponds to the sum of the single stage cost for the given pair and the discounted optimal cost-to-go from the next state onwards. The Q-learning algorithm has become a popular reinforcement learning technique and has been applied in several settings.

Amongst other reinforcement learning algorithms, temporal difference learning (Bertsekas & Tsitsiklis, 1996) is also based on the value iteration technique (however, it only performs policy evaluation and not control) while actor-critic algorithms (Konda & Borkar, 1999) are based on policy iteration. In classical value iteration, one starts with a given initial value function and uses a fixed point iteration to solve the Bellman equation. Unlike classical value iteration, temporal difference and Q-learning algorithms do not require transition probability information.

In another body of research, multi-timescale stochastic approximation has been studied recently (Borkar, 1997). Such algorithms have been developed for a variety of settings such as simulation optimization (Bhatnagar, Fu, Marcus, & Wang, 2003) and reinforcement learning (Konda & Borkar, 1999). The main idea in these algorithms is to use multiple step-size schedules or timescales that have different rates of convergence. Recursions governed by step-sizes that tend to zero slower converge faster because of their better tracking abilities.

☆ This paper was not presented at any IFAC meeting. This paper was recommended for publication in revised form by Associate Editor George Yin under the direction of Editor Ian Petersen.

* Corresponding author.
E-mail addresses: shalabh@csa.iisc.ernet.in (S. Bhatnagar), mohanbabu@motorola.com (K.M. Babu).

0005-1098/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.automatica.2007.09.009

This approach has resulted in the development of computationally superior algorithms in the case of schemes that typically involve two nested loops with an outer loop update being performed once after convergence of the inner loop procedure. Another significant advance in recent years has been the development of simultaneous perturbation stochastic approximation (SPSA) methodology for gradient estimation (Spall, 1992, 1997). A one-simulation SPSA-based gradient estimation procedure is proposed in Spall (1997) that however does not yield good performance unlike its two-simulation counterpart in Spall (1992). Multi-timescale simulation optimization algorithms based on SPSA are presented in Bhatnagar et al. (2003). Certain deterministic constructions for the perturbation sequences based on Hadamard matrices are found to considerably improve performance when one-simulation SPSA estimates are used.

In this paper, we develop two Q-learning-based algorithms that (both) use two-timescale stochastic approximation. The first of these updates Q-values of all feasible state–action pairs at each instant while the other involves a partly asynchronous implementation and updates Q-values of states with actions chosen according to 'current' randomized policy updates. The regular Q-learning algorithm involves computation of an action that attains the minimum value amongst current Q-value updates. The above minimum is obtained prior to the averaging step. Our algorithms obtain minimum in the space of stationary randomized policies using a gradient search recursion based on one-simulation SPSA with Hadamard matrix-based construction (Bhatnagar et al., 2003). This is done on the faster timescale and on the slower one, averaging of Q-values is performed. Our method may allow for more efficient implementations such as least squares or second order methods as compared to regular Q-learning because of the use of gradient search in the algorithm updates. Moreover, since we use stationary randomized policy updates for picking the actions that minimize Q-values, other actions get picked as well with certain (albeit diminishing) probabilities obtained from the above policy updates. For regular Q-learning, an additional exploration step is recommended whereby one picks 'non-minimizing' actions with certain small probabilities (Bertsekas & Tsitsiklis, 1996). This is taken care of in a natural manner using randomized policies in our algorithms. We give a proof of convergence of the algorithms. Next, we consider an application of our algorithms on a problem of routing packets in communication networks. We show results of numerical experiments on a few different settings and present performance comparisons with the Q-learning algorithm. Both our algorithms exhibit fast convergence and give the corresponding optimal policies in addition to Q-values. Our second algorithm may have computational advantages in scenarios where the number of feasible actions in each state is large. We do not consider in this paper a setting involving feature-based state representations even though this can be easily incorporated.

The rest of the paper is organized as follows. The framework and algorithms are presented in Section 2. The convergence analysis of the algorithms is provided in Section 3. Numerical experiments over different network topologies for a problem of routing are presented in Section 4. Finally, Section 5 presents the concluding remarks.

2. Framework and algorithms

Suppose $\{X_n\}$ is a controlled Markov chain or a Markov decision process (MDP) taking values in a set $S=\{1, 2, \ldots, s\}$ with $s<\infty$. Let $U(i)$ denote the set of all feasible actions or controls in state $i \in S$. We assume that $U(i)$ are finite sets and in particular have the form $U(i)=\{u_i^0, u_i^1, \ldots, u_i^N\}$. Note that we assume only for notational simplicity that each feasible action set $U(i)$ has exactly $(N+1)$ elements. In general, N could also be a function of state i. Suppose $p(i, u, j)$ and $g(i, u, j), i, j \in S, u \in U(i)$, respectively, denote the one-step transition probability and the single-stage cost when the MDP is in state i and a feasible action u is chosen, and the next state is j. We assume that $g(i, u, j)$ are nonnegative and bounded. Let $U=\bigcup_{i\in S}U(i)$ denote the set of all possible actions. By an admissible policy $\bar{\psi}$, we mean a sequence of functions $\bar{\psi}=\{\mu_0, \mu_1, \mu_2, \ldots\}$ with each $\mu_k : S \to U$ such that $\mu_k(i) \in U(i), i \in S, k \geqslant 0$. Let Ψ be the set of all admissible policies. If $\mu_k \triangleq \mu, \forall k \geqslant 0$, then we call $\bar{\psi}=\{\mu, \mu, \ldots\}$ or by abuse of notation, the function μ itself as a stationary deterministic policy or simply a stationary policy.

A randomized policy $\hat{\psi}$ is a sequence $\hat{\psi}=\{\phi_1, \phi_2, \ldots\}$ of distributions $\phi_n : S \to \mathbf{P}(U), n \geqslant 1$ with $\mathbf{P}(U)$ being the set of all probability vectors on U and, in particular, for each $i \in S$, $n \geqslant 1$, $\phi_n(i) \in \mathbf{P}(U(i))$, where $\mathbf{P}(U(i))$ is the set of all probability vectors on $U(i)$. A stationary randomized policy is one for which $\phi_n(i) \triangleq \phi(i)\ \forall n \geqslant 1$. In what follows, we shall use π_i to denote the vector of probabilities $\pi_i=(\pi_i(u_i^0), \ldots, \pi_i(u_i^N))$ with $\pi_i(u)$ being the probability of picking action u in state i. Thus $\pi=(\pi_1, \ldots, \pi_s)$ shall represent (by an abuse of notation) a stationary randomized policy.

Let $\alpha \in (0, 1)$ be a given constant. The aim is to minimize over all admissible policies $\bar{\psi}=\{\mu_0, \mu_1, \mu_2, \ldots\}$, the infinite horizon α-discounted cost

$$J_{\bar{\psi}}(i)=\lim_{T\to\infty} E\left[\sum_{k=0}^{T}\alpha^k g(X_k, \mu_k(X_k), X_{k+1})|X_0=i\right]. \quad (1)$$

Let

$$J^*(i)=\min_{\bar{\psi}\in\Psi} J_{\bar{\psi}}(i), \quad i \in S \quad (2)$$

denote the optimal cost. For a given stationary policy μ, $J_\mu(.)$ is called the value function corresponding to policy μ. One can show that an optimal stationary policy exists here and J^* satisfies the Bellman equation

$$J^*(i)=\min_{u\in U(i)}\left(\sum_{j\in S} p(i, u, j)(g(i, u, j)+\alpha J^*(j))\right). \quad (3)$$

From the above, an optimal stationary randomized policy can be seen to exist as well. Such a policy would assign the entire probability mass to the optimal action in each state provided such an action is unique. Else, it would assign equal non-zero

probabilities to all optimizing actions and zero probabilities to the rest. The class of stationary randomized policies thus contains within itself the class of stationary policies. In fact, stationary policies correspond to vertices in the space of stationary randomized policies, the latter being a closed convex hull of the former.

For given $i \in S, u \in U(i)$, define Q-values $Q^*(i,u)$ (Bertsekas & Tsitsiklis, 1996) by

$$Q^*(i,u) = \sum_{j \in S} p(i,u,j)(g(i,u,j) + \alpha J^*(j)). \tag{4}$$

The Bellman equation now corresponds to

$$J^*(i) = \min_{u \in U(i)} Q^*(i,u), \quad i \in S. \tag{5}$$

Substituting (5) in (4) gives

$$Q^*(i,u) = \sum_{j \in S} p(i,u,j)\left(g(i,u,j) + \alpha \min_{v \in U(j)} Q^*(j,v)\right). \tag{6}$$

The regular Q-learning algorithm is a stochastic approximation version of (6) and proceeds as follows:

$$Q_{n+1}(i,u) = Q_n(i,u) + \gamma(n)(g(i,u,\eta_n(i,u)) + \alpha \min_{v \in U(\eta_n(i,u))} Q_n(\eta_n(i,u),v) - Q_n(i,u)). \tag{7}$$

In the above, $\eta_n(i,u)$ is a simulated next state when current state is i and action $u \in U(i)$ is chosen. It is assumed that $\eta_n(i,u), n \geqslant 0$, are independent random variables each having distribution $p(i,u,\cdot)$. Also, $\{\gamma(n)\}$ is a step-size sequence that satisfies $\gamma(n) > 0\ \forall n \geqslant 0$,

$$\sum_n \gamma(n) = \infty \quad \text{and} \quad \sum_n \gamma(n)^2 < \infty.$$

In what follows, we present two variants of (7) that both use two-timescale stochastic approximation. In these algorithms, explicit minimization in (7) is avoided and instead a gradient search over stationary randomized policies is performed on a faster timescale. As stated before, the search space of stationary randomized policies in our algorithms corresponds to the closed convex hull of the corresponding space of stationary deterministic policies. In regular Q-learning, only stationary deterministic policies are used as a result of which it suffers from the problem of *lack of sufficient exploration* (Bertsekas & Tsitsiklis, 1996). This problem gets worse when the numbers of states and actions are large. An (additional) exploration step is usually recommended whereby actions that do not correspond to the 'minimizing action' are picked with some small probability. With our algorithms, on the other hand, an additional exploration step is not required since actions that do not minimize Q-value updates automatically get picked with probabilities prescribed by the randomized policy updates.

While our first algorithm updates Q-values for all state–action pairs at each instant, the second updates Q-values for pairs of states with actions picked according to current randomized policy updates. In what follows, we shall represent a randomized policy as $\pi = (\hat{\pi}_1, \ldots, \hat{\pi}_s)$ with each $\hat{\pi}_i = (\pi_i(u), u \in U(i) \backslash \{u_i^0\})$, $i \in S$. Note that $\pi_i(u_i^0)$, $i \in S$ are directly obtained from the components of $\hat{\pi}_i$ as $\pi_i(u_i^0) = 1 - \sum_{j=1}^N \pi_i(u_i^j)$. Let $PS \subset \mathbf{R}^N$ denote the simplex

$$PS = \left\{(y_1, \ldots, y_N) | y_i \geqslant 0, 1 \leqslant i \leqslant N \text{ and } \sum_{i=1}^N y_i \leqslant 1\right\},$$

in which $\hat{\pi}_i, i \in S$, take values. Let $\Gamma : \mathbf{R}^N \to PS$ denote the projection map. Consider now $\{\pm 1\}^N$-valued vectors $\Delta_n(i) = (\Delta_n(i,u_i^1), \ldots, \Delta_n(i,u_i^N))$, $i \in S$, that are used to perturb policy updates so as to obtain suitable gradient estimates of the Q-function. These are obtained in a manner explained below (Bhatnagar et al., 2003).

2.1. Construction for perturbation sequences $\Delta_n(i)$

Let H_P be a normalized Hadamard matrix (a Hadamard matrix is said to be normalized if all the elements of its first row and column are 1s) of order P with $P \geqslant N+1$. Let $h(1), \ldots, h(N)$ be any N columns other than the first column of H_P, and form a new $P \times N$ dimensional matrix H'_P which has the above as its columns. Let $\Delta(p)$, $p = 1, \ldots, P$ be the P rows of H'_P. Now set $\Delta_n(i) = \Delta(n \bmod P + 1), \forall n \geqslant 0, i \in S$. The perturbations are thus generated by cycling through the rows of H'_P. Here P is chosen as $P = 2^{\lceil \log_2 (N+1) \rceil}$. Finally, matrices H_P for $P = 2^k$, $k \geqslant 1$ are systematically constructed as follows:

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \text{and} \quad H_{2^k} = \begin{pmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{pmatrix}, \quad k > 1.$$

The perturbations obtained above are seen to satisfy the desired conditions in Theorem 2.5 of Bhatnagar et al. (2003), see Lemma 3.5 there. Hence one obtains the correct gradient direction by using these sequences.

2.2. Algorithm-1

Let $Q_n(.,.)$ and $\hat{\pi}_i(n)$, $i \in S$ denote the nth updates of the Q-value function and randomized policy, respectively. Let $\pi'_i(n) = \Gamma(\hat{\pi}_i(n) - \delta\Delta_n(i))$ where $\delta > 0$ is a given small constant. Let $\{a(n)\}$ and $\{b(n)\}$ be two step-size sequences that satisfy $a(n), b(n) > 0, \forall n \geqslant 0$ and

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

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

respectively. Further, $a(n+1)/a(n), b(n+1)/b(n) \to 1$ as $n \to \infty$. Let $(\Delta_n(i))^{-1} \triangleq (1/\Delta_n(i,u_i^1), \ldots, 1/\Delta_n(i,u_i^N))^{\mathrm{T}}$, $\forall i \in S$. Let $\psi_n(j)$ denote the action chosen from the set $U(j)$ according to the distribution given by $\pi'_j(n)$, with probability of picking action $u_j^0 \in U(j)$ automatically specified from the latter. Also, let $\eta_n(i,u)$ be as in (7).

Now for all $i \in S, u \in U(i)$, initialize $Q_0(i,u) = 0$ and $\hat{\pi}_i(0) = (1/N+1, \ldots, 1/N+1)$, respectively. Any other choice

of $\hat{\pi}_i(0)$ may also be used. Then $\forall i \in S, u \in U(i)$,

$$Q_{n+1}(i,u) = Q_n(i,u) + b(n)(g(i,u,\eta_n(i,u)) + \alpha Q_n(\eta_n(i,u), \psi_n(\eta_n(i,u))) - Q_n(i,u)), \quad (9)$$

$$\hat{\pi}_i(n+1) = \Gamma\left(\hat{\pi}_i(n) + a(n)\frac{Q_n(i,\psi_n(i))}{\delta}(\Delta_n(i))^{-1}\right). \quad (10)$$

2.3. Algorithm-2

This is similar to Algorithm-1, except that we do not update Q-values for each state–action pair as in (9). Instead, Q-value updates are performed for states with actions picked according to the 'current' randomized policy update. Also, recursion (10) is exactly the same as before. Thus, (9) is replaced here by the following recursion:

$$\begin{aligned} Q_{n+1}(i,\hat{\psi}_n(i)) = {} & (1-b(n))Q_n(i,\hat{\psi}_n(i)) \\ & + b(n)(g(i,\hat{\psi}_n(i),\eta_n(i,\hat{\psi}_n(i))) \\ & + \alpha Q_n(\eta_n(i,\hat{\psi}_n(i)), \psi_n(\eta_n(i,\hat{\psi}_n(i))))). \end{aligned} \quad (11)$$

In the above, $\hat{\psi}_n(i)$ is the action chosen from the set $U(i)$ according to the distribution $\hat{\pi}_i(n)$, with the probability of picking action $u_i^0 \in U(i)$ automatically specified from the latter. This corresponds to an asynchronous implementation of Algorithm-1 whereby at each instant, an action for each state is first selected (according to the given randomized policy update) and then the corresponding Q-value is updated.

3. Convergence analysis

We first consider Algorithm-1. Let $\mathbf{F}_n = \sigma(Q_j(i,u), \eta_j(i,u), \hat{\pi}_i(j),\ j \leqslant n, \psi_j(i),\ j < n, u \in U(i), i \in S)$ denote an increasing sequence of σ-fields. One can show in a similar manner as Theorem 2.1 of Tsitsiklis (1994) that $\sup_n \|Q_n(i,u)\| < \infty \forall i \in S, u \in U(i)$. Define sequences $\{M_n(i)\}, i \in S$, according to

$$M_n(i) = \sum_{j=0}^{n-1} a(j)(Q_j(i,\psi_j(i)) - E[Q_j(i,\psi_j(i))|\mathbf{F}_j]).$$

Then $\{M_n(i)\}$ can be seen to be martingale sequences that are almost surely convergent (since their associated quadratic variation processes converge because of (8)). Hence the faster timescale recursion (10) can be written as

$$\hat{\pi}_i(n+1) = \Gamma\left(\hat{\pi}_i(n) + a(n)\frac{E[Q_n(i,\psi_n(i))|\mathbf{F}_n]}{\delta} \times (\Delta_n(i))^{-1} + \varepsilon_1(n)\right), \quad (12)$$

where $\varepsilon_1(n) = o(1)$ by the above. Further, recursion (9) can be written as

$$Q_{n+1}(i,u) = Q_n(i,u) + a(n)\varepsilon_2(n), \quad (13)$$

where, since $b(n) = o(a(n))$ by (8), $\varepsilon_2(n) \to 0$ as $n \to \infty$. Thus, when viewed from the timescale corresponding to $\{a(n)\}$, recursion (9) can be seen to asymptotically track the trajectories of the ordinary differential equation (ODE):

$$\dot{Q}_t(i,u) = 0, \quad i \in S, \ \ u \in U(i). \quad (14)$$

Note that by (14), $Q_t(i,u)$ are time-invariant when viewed from the faster timescale; hence we suppress the index t and denote by $Q(i,u)$ the above. Now, in (12), suppose that $Q_j^{\pi_i'(j)}(i) \triangleq E[Q_j(i,\psi_j(i))|\mathbf{F}_j]$. By using a Taylor's expansion of $Q_j^{\pi_i'(j)}(i)$ around the point $\hat{\pi}_i(j)$ and using similar arguments as in Corollary 2.6 of Bhatnagar et al. (2003), one can see that (12) asymptotically tracks the trajectories of the following ODE, in the limit as $\delta \to 0$.

$$\dot{\hat{\pi}}_i = \hat{\Gamma}(-\nabla Q^{\hat{\pi}_i(t)}(i)), \quad (15)$$

where $\hat{\Gamma}(.)$ is defined according to

$$\hat{\Gamma}(v(y)) = \lim_{\gamma \downarrow 0}\left(\frac{\Gamma(y+\gamma v(y)) - \Gamma(y)}{\gamma}\right),$$

for any bounded, continuous $v(.)$. The stable fixed points of (15) lie within the set $M = \{\hat{\pi}_i | \hat{\Gamma}(\nabla Q^{\hat{\pi}_i}(i)) = 0\}$. It can be seen that $V_{\hat{\pi}} = \sum_{i \in S} Q^{\hat{\pi}_i}(i)$ serves as a strict Liapunov function for the ODE (15). Hence, let $\pi_i^u(n) \to \pi_i^{u,*}$ as $n \to \infty$. Note that if u^* corresponds to a unique optimal action in state i, then the following will be true: $\pi_i^{u,*} = 1$ for $u = u^*$ and $\pi_i^{u,*} = 0$ for all $u \in U(i)$, $u \neq u^*$. If on the other hand, the optimal action is not unique, then one expects policy π^* in state i to assign equal positive mass to all optimal actions (viz., those whose Q-values are equal and are uniformly lower compared to those of other actions).

Let $Q^*(i,u)$, $i \in S, u \in U(i)$ correspond to the unique solution of (4) and consider the slower timescale recursion (9). We obtain

Theorem 1. *For all $i \in S, u \in U(i)$, the quantities $Q_n(i,u)$ as given by Algorithm-1 converge almost surely to $Q^*(i,u)$ in the limit as $\delta \to 0$.*

Proof. Consider σ-fields $\mathbf{G}_n = (Q_j(i,u), \hat{\pi}_i(j),\ j \leqslant n; \eta_j(i,u), \psi_j(i),\ j < n,\ i \in S,\ u \in U(i)),\ n \geqslant 1$. For given $i \in S$, $u \in U(i)$, define

$$\begin{aligned} N_n(i,u) = \sum_{j=0}^{n-1} b(j)[(g(i,u,\eta_j(i,u)) & \\ + \alpha Q_j(\eta_j(i,u), \psi_j(\eta_j(i,u)))) - E[(g(i,u,\eta_j(i,u)) & \\ + \alpha Q_j(\eta_j(i,u), \psi_j(\eta_j(i,u))))|\mathbf{G}_j]]. & \end{aligned}$$

Now since $\sup_n |Q_n(i,u)| < \infty$ for all $i \in S$, $u \in U(i)$ and $\sum_n b(n)^2 < \infty$, one can again see that $\{N_n(i,u)\}$ are almost surely convergent martingale sequences. Now observe that

$$\begin{aligned} & E[(g(i,u,\eta_j(i,u)) + \alpha Q_j(\eta_j(i,u), \psi_j(\eta_j(i,u))))|\mathbf{G}_j] \\ & = \sum_{k \in S} p(i,u,k)\left[g(i,u,k) + \alpha \sum_{v \in U(k)} \pi_k^{v,*} Q_j(k,v)\right]. \end{aligned} \quad (16)$$

By the foregoing, recursion (9) can be rewritten as

$$Q_{n+1}(i,u) = Q_n(i,u) + b(n)\sum_{k\in S} p(i,u,k) \times \left[g(i,u,k) + \alpha \sum_{v\in U(k)} \pi_k^{v,*} Q_n(k,v) \right] + \zeta(n) - b(n) Q_n(i,u), \tag{17}$$

where $\zeta(n) = (N_{n+1}(i,u) - N_n(i,u)) \to 0$ as $n \to \infty$ with probability one. Recursion (17) is then a 'noisy' Euler discretization of the ODE

$$\dot{Q}(i,u) = \sum_{k\in S} p(i,u,k) \left[g(i,u,k) + \alpha \sum_{v\in U(k)} \pi_k^{v,*} Q(k,v) \right] - Q(i,u). \tag{18}$$

For the above ODE, the unique asymptotically stable equilibrium corresponds to the solution to the following system of equations.

$$\bar{Q}(i,u) = \sum_{k\in S} p(i,u,k) \left[g(i,u,k) + \alpha \sum_{v\in U(k)} \pi_k^{v,*} \bar{Q}(k,v) \right]. \tag{19}$$

Note that we have plugged in $\pi_k^{v,*}$ directly in the expression for the expectation in (16) and the subsequent analysis. This corresponds to the case of $\delta \to 0$ in the faster scale recursion (10). Ordinarily, since $\delta > 0$ is held fixed in the algorithm, one should have $\pi_k^{v,\delta}$ in place of the above as the stationary probabilities for given Q-value updates. However, it is easy to see by writing the corresponding faster scale ODE for fixed $\delta > 0$ that $\pi_k^{v,\delta} \to \pi_k^{v,*}$ as $\delta \to 0$, as the trajectories of the above ODE converge to those of (15) uniformly on compacts for the same initial condition in both. Likewise, if $\bar{Q}^\delta(i,u)$ corresponds to the solution of an analogous equation as (19) with $\pi_k^{v,\delta}$ in place of $\pi_k^{v,*}$ (with a corresponding change in ODE (18) as well), it can again be seen that the trajectories of such an ODE would converge to those of (18) for the same initial condition, uniformly on compacts. Hence, $\bar{Q}^\delta(i,u)$ would converge as $\delta \to 0$ to the solution of (19).

Now it is easy to see that $Q^*(i,u)$ is the unique solution to (19). The iterates $Q_n(i,u)$ can now be shown via standard arguments (Borkar, 1997) to converge with probability one to $Q^*(i,u)$. This completes the proof. □

Finally, note that a similar analysis can be shown for Algorithm-2 as well. Recall from the previous discussion that (11) constitutes asynchronous updates in which the actions $\hat{\psi}_n(i)$ (in the tuples $(i, \hat{\psi}_n(i))$ whose Q-values are updated) are sampled according to running policy updates $\hat{\pi}_i(n)$. The algorithm can be seen to converge in a similar manner as Algorithm-1 to the pair of optimal policy and Q-values.

4. Numerical experiments

Efficiently routing data is an important decision making problem in communication networks. This is all the more important as networks are subject to frequent and unpredictable changes in topology and link costs. Conventional internet routing algorithms such as the routing information protocol (RIP) or open shortest path first (OSPF) are based on minimizing the number of hops which means number of relay nodes between the source and destination. With every change in topology, the network generates several routing information packets and a lot of time is needed to arrive at optimal paths. Reinforcement learning-based methods have recently been applied for routing in communication networks. For instance, Marbach, Mihatsch, and Tsitsiklis (2000) uses temporal difference learning for routing in integrated service networks. In Boyan and Littman (1994), a 'Q-routing' algorithm based on Q-learning is proposed.

We study the performance of our algorithms for the problem of routing in communication networks. For our experiments, we consider networks with different topologies involving 4, 8 and 16 nodes, respectively. We show here results of experiments with networks having 4 and 16 nodes only as similar results were obtained for the 8-node case. For simplicity, we denote by A-1 and A-2, our algorithms 1 and 2, respectively. Also, Q-L shall denote the Q-learning algorithm. We also analytically compute the Q-values on these settings. These are denoted A-Q. In all cases studied, 0 is the source and the node with highest number is the destination. The network configurations are shown in Figs. 1 and 2, respectively.

We assume that all links are bidirectional and have the same cost values along both directions. For ease of exposition, we assign numbers 0, 1, 2, etc., to the links emerging from the various nodes, see Figs. 1 and 2. Thus, for instance in Fig. 2, link 0 at node 4 corresponds to the same link as link 2 at node 1 (but in the reverse direction). The action set $U(i)$ at node i corresponds to the set of links connected to that node. Selecting an action at a node thus corresponds to selecting a link at that node for routing packets. We denote by $(n_1 - n_2 - \cdots - n_{l-1} - n_l)$ a path from node n_1 to node n_l through intermediate nodes $n_2, \ldots, n_{l-1}$. We choose $\alpha = 0.9$, $\delta = 0.06$, $a(n) = 1/n$, $b(n) = 1/n^{0.7}$, $\forall n \geqslant 1$, with $a(0) = b(0) = 1$, in all our

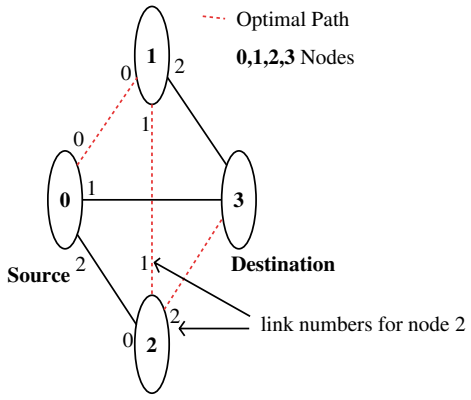

Fig. 1. Network with 4 nodes.

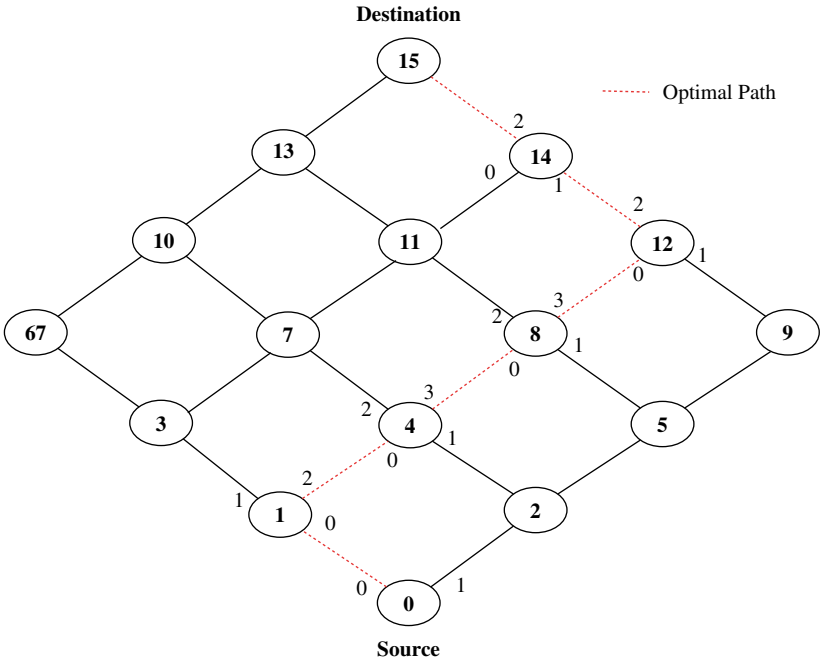

Fig. 2. Network with 16 nodes.

Table 1
Converged probability vectors on the optimal path (0-1-2-3)

Alg.	Node (x)	$p(x,0)$	$p(x,1)$	$p(x,2)$
A-1	0	0.81	0.11	0.08
	1	0.21	0.53	0.26
	2	0.02	0.00	0.97
A-2	0	0.91	0.01	0.07
	1	0.15	0.84	0.01
	2	0.01	0.13	0.86

Table 2
Converged Q-values for nodes on the optimal path (0-1-2-3)

Alg.	Node (x)	$Q(x,0)$	$Q(x,1)$	$Q(x,2)$
A-1	0	0.55	1.12	1.00
	1	0.69	0.22	1.00
	2	1.59	0.54	0.10
A-2	0	0.35	1.15	0.98
	1	0.48	0.22	0.89
	2	1.26	0.34	0.10
Q-L	0	0.27	1.09	1.00
	1	0.34	0.19	1.00
	2	1.24	0.27	0.100
A-Q	0	0.35	1.10	1.00
	1	0.35	0.20	1.00
	2	1.20	0.45	0.10

experiments. When the one-stage costs are high in magnitude, normalizing these is seen to improve performance. The network learns quickly to route packets along the shortest path. The costs on each link at an instant are chosen according to a uniform distribution over an interval. The latter (interval) for each link is selected in a manner that the optimal paths get specified. For both settings below, we run all three algorithms (A-1, A-2 and Q-L) for 50,000 updates. We observe however that convergence for all three algorithms is achieved in much less number of updates (less than 5000 updates in all cases studied).

Table 3
Converged probability vectors for nodes on the optimal path (0-1-4-8-12-14-15) using A-1 and A-2

Alg.	Node (x)	$p(x,0)$	$p(x,1)$	$p(x,2)$	$p(x,3)$
A-1	0	0.96	0.04	–	–
	1	0.05	0.32	0.63	–
	4	0.11	0.40	0.06	0.43
	8	0.01	0.13	0.37	0.49
	12	0.13	0.17	0.70	–
	14	0.03	0.01	0.96	–
A-2	0	0.90	0.10	–	–
	1	0.03	0.26	0.71	–
	4	0.22	0.02	0.28	0.48
	8	0.07	0.09	0.11	0.73
	12	0.20	0.03	0.77	–
	14	0.02	0.00	0.98	–

4.1. Network with 4 nodes

We consider here a network (see Fig. 1) with four nodes and six bidirectional links. The results obtained for the various algorithms when the optimal path corresponds to (0-1-2-3) are shown in Tables 1 and 2, respectively. Here $p(\cdot,\cdot)$ and $Q(\cdot,\cdot)$ denote the converged randomized policy and Q-value obtained for the node-link pair $(\cdot,\cdot)$. We observed that convergence to

Table 4
Converged Q-values for nodes on the optimal path (0-1-4-8-12-14-15) using A-1, A-2, Q-L and A-Q

Alg.	Node (x)	$Q(x,0)$	$Q(x,1)$	$Q(x,2)$	$Q(x,3)$
A-1	0	0.12	0.16	–	–
	1	0.11	0.16	0.12	–
	4	0.12	0.16	0.15	0.08
	8	0.11	0.14	0.10	0.06
	12	0.086	0.12	0.05	–
	14	0.10	0.07	0.00	–
A-2	0	0.24	0.35	–	–
	1	0.40	0.34	0.23	–
	4	0.25	0.63	0.31	0.17
	8	0.30	0.30	0.30	0.15
	12	0.19	0.38	0.13	–
	14	0.26	0.15	0.00	–
Q-L	0	0.47	2.18	–	–
	1	0.52	2.23	0.41	–
	4	0.47	2.18	2.18	0.34
	8	0.41	2.12	1.98	0.27
	12	0.34	2.05	0.19	–
	14	1.98	0.27	0.10	–
A-Q	0	0.36	1.50	–	–
	1	0.40	1.54	0.30	–
	4	0.40	1.48	1.48	0.24
	8	0.35	1.44	1.39	0.20
	12	0.24	1.38	0.14	–
	14	1.26	0.20	0.10	–

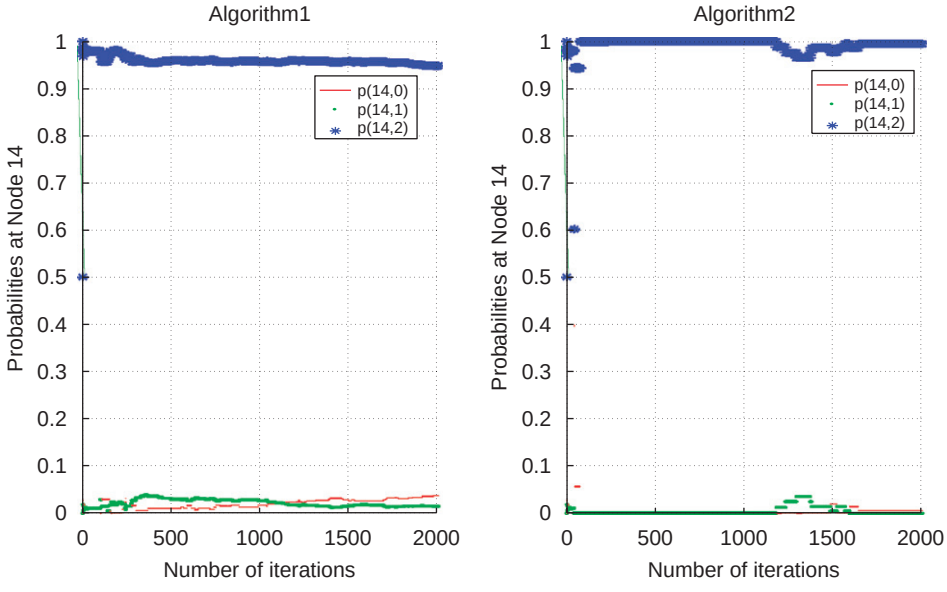

Fig. 3. Plot of probability updates vs. number of iterations at node 14 for optimal path (0-1-4-8-12-14-15).

the optimal paths is achieved somewhat faster when A-2 is used as compared to A-1.

4.2. Network with 16 nodes

We consider here a network with 16 nodes and 24 bidirectional links, see Fig. 2. The results obtained for the optimal path setting (0-1-4-8-12-14-15) using the various algorithms are shown in Tables 3 and 4, respectively.

We show in Fig. 3, the plots of policy updates vrs. the number of iterations using our algorithms A-1 and A-2. Also, Fig. 4 provides similar plots of Q-value updates. On a Pentium IV computer with 2.4 GHz processor, it took less than 1 min to complete one simulation run of 50,000 iterations for all three

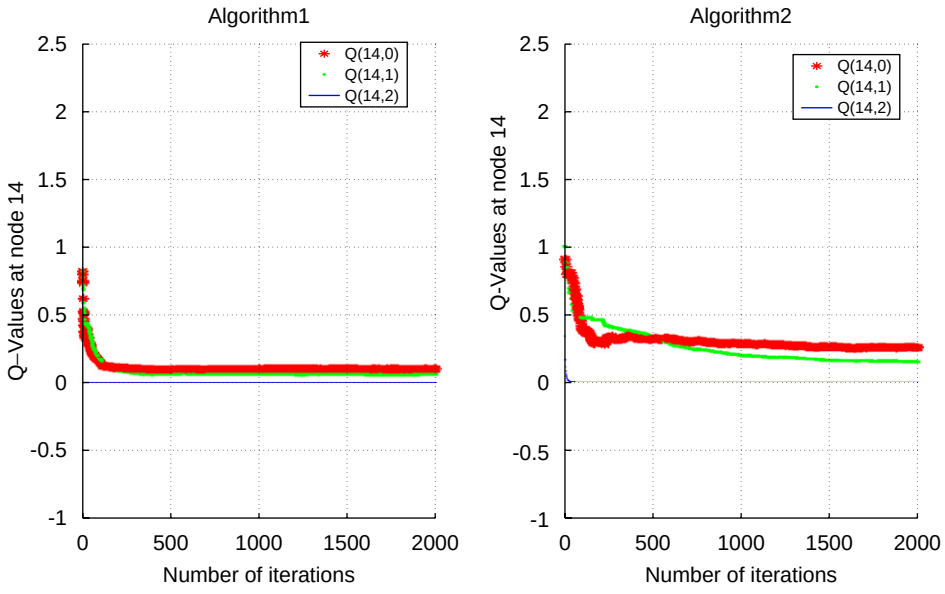

Fig. 4. Plot of Q-factors vs. number of iterations at node 14 for optimal path (0-1-4-8-12-14-15).

algorithms (A-1, A-2 and Q-L) in all cases studied. The codes for the various network topologies were written using the C programming language. We also observed convergence to optimal path configurations using our algorithms under different link costs.

5. Conclusions

In this paper, we developed two gradient search-based variants of the Q-learning algorithm that both use two-timescale stochastic approximation. We studied applications of these to the problem of routing in communication networks. One of the algorithms (Algorithm-1) updates Q-values associated with each state–action pair while the other (Algorithm-2) updates Q-values of states with actions chosen according to 'current' randomized policy updates. Both our algorithms pick 'minimizing actions' for the Q-value updates using probabilities obtained from the randomized policy updates. Thus the 'non-minimizing' actions are also picked with certain, albeit diminishing, probabilities. For regular Q-learning, an additional exploration step is recommended whereby one picks 'non-minimizing' actions with certain small probabilities (Bertsekas & Tsitsiklis, 1996). This is taken care of in a natural manner using randomized policies in our algorithms. We gave a proof of convergence of the algorithms. Both our algorithms are found to converge to the optimal path configurations. Algorithm-2, however, may have computational advantages in scenarios where the numbers of states and actions are large.

We considered synchronous implementations for regular Q-learning and Algorithm-1, and a partly asynchronous implementation for Algorithm-2. Fully asynchronous and distributed implementations (Tsitsiklis, 1994) for the algorithms may also be tried. Further, in the settings that we considered, the associated MDP had deterministic state transitions given actions. It would be helpful to consider also MDPs with non-deterministic transitions for comparing performance of the proposed algorithms with Q-learning. Also, one could consider settings with dynamically varying topologies and link costs as with real network scenarios. For large state and action spaces, suitable modifications to our algorithms that incorporate feature-based representations for state and action spaces, and parameterizations of Q-values and policies need be derived. The performance of the modified algorithms may then be studied in implementations involving large scale networks. This is a possible future direction.

Acknowledgments

This work was supported in part by Grants SR/S3/EE/43/2002-SERC-Engg and SR/S3/EECE/011/2007 from the Department of Science and Technology, Government of India.

References

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Belmont, MA: Athena Scientific.

Bhatnagar, S., Fu, M. C., Marcus, S. I., & Wang, I.-J. (2003). Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences. *ACM Transactions on Modelling and Computer Simulation*, *13*(2), 180–209.

Borkar, V. S. (1997). Stochastic approximation with two time scales. *Systems Control Letters*, *29*, 291–294.

Boyan, J. A., & Littman, M. L. (1994). Packet routing in dynamically changing networks: A reinforcement learning approach. *Advances in Neural Information Processing Systems*, *6*, 671–678.

Konda, V. R., & Borkar, V. S. (1999). Actor-critic like learning algorithms for Markov decision processes. *SIAM Journal on Control and Optimization*, *38*(1), 94–123.

Marbach, P., Mihatsch, O., & Tsitsiklis, J. N. (2000). Call admission control and routing in integrated services networks using neuro-dynamic programming. *IEEE Journal on Selected Areas in Communication*, *18*, 197–208.

Spall, J. C. (1992). Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, *37*, 332–341.

Spall, J. C. (1997). A one-measurement form of simultaneous perturbation stochastic approximation. *Automatica*, *33*, 109–112.

Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and Q-learning. *Machine Learning*, *16*, 185–202.

Watkins, J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279–292.

Shalabh Bhatnagar received his Ph.D. in Electrical Engineering from the Indian Institute of Science, Bangalore, in 1998. He holds a permanent position as Associate Professor in the Department of Computer Science and Automation at the Indian Institute of Science, Bangalore.
He has held visiting research appointments at the University of Maryland, College Park, MD, USA, Free University, Amsterdam, Netherlands, Indian Institute of Technology, Delhi and University of Alberta, Canada.
His research interests are in stochastic approximation algorithms, Markov decision processes, reinforcement learning and communication networks. He is an Associate Editor for IEEE Transactions on Automation Science and Engineering and is a Senior Member, IEEE.

K. Mohan Babu received his Bachelors degree in Electrical and Electronics Engineering and Masters Degree in Systems Science and Automation from the Indian Institute of Science (IIS), Bangalore in 2005. Since then he has been with Motorola. Currently, he is involved in the area of Network Advanced Technologies.