**April 2010** **Report LIDS - 2831**

Also as: Tech. Report C-2010-10
Univ. of Helsinki

# Q-Learning and Enhanced Policy Iteration in Discounted Dynamic Programming

**Dimitri P. Bertsekas**[1] **and Huizhen Yu**[2]

**Abstract**

We consider the classical finite-state discounted Markovian decision problem, and we introduce a new policy iteration-like algorithm for finding the optimal Q-factors. Instead of policy evaluation by solving a linear system of equations, our algorithm requires (possibly inexact) solution of a nonlinear system of equations, involving estimates of state costs as well as Q-factors. This is Bellman's equation for an optimal stopping problem that can be solved with simple Q-learning iterations, in the case where a lookup table representation is used; it can also be solved with the Q-learning algorithm of Tsitsiklis and Van Roy [TsV99], in the case where feature-based Q-factor approximations are used. In exact/lookup table representation form, our algorithm admits asynchronous and stochastic iterative implementations, in the spirit of asynchronous/modified policy iteration, with lower overhead and more reliable convergence advantages over existing Q-learning schemes. Furthermore, for large-scale problems, where linear basis function approximations and simulation-based temporal difference implementations are used, our algorithm resolves effectively the inherent difficulties of existing schemes due to inadequate exploration.

## 1. INTRODUCTION

We consider the approximate solution of large-scale discounted infinite horizon dynamic programming (DP) problems. The states are denoted $i = 1, \ldots, n$. State transitions $(i, j)$ under control $u$ occur at discrete times according to given transition probabilities $p_{ij}(u)$, and generate a cost $\alpha^k g(i, u, j)$ at time $k$, where $\alpha \in (0, 1)$ is a discount factor. We consider deterministic stationary policies $\mu$ such that for each $i$, $\mu(i)$ is a control that belongs to a constraint set $U(i)$. We denote by $J_\mu(i)$ the total discounted expected cost of $\mu$ over an infinite number of stages starting from state $i$, and by $J^*(i)$ the minimal value of $J_\mu(i)$ over all $\mu$. We denote by $J_\mu$ and $J^*$ the vectors of $\Re^n$ ($n$-dimensional space) with components $J_\mu(i)$ and $J^*(i)$, $i = 1, \ldots, n$, respectively. This is the standard discounted Markovian decision problem (MDP) context, discussed in many sources (e.g.,

[1] Dimitri Bertsekas is with the Dept. of Electr. Engineering and Comp. Science, M.I.T., Cambridge, Mass., 02139, dimitrib@mit.edu. His research was supported by NSF Grant ECCS-0801549.

[2] Huizhen Yu is with the Dept. of Computer Science, Univ. of Helsinki, Finland, janey.yu@cs.helsinki.fi. Her research was supported in part by Academy of Finland Grant 118653 (ALGODAN) and the PASCAL Network of Excellence, IST-2002-506778.

Bertsekas [Ber07], Puterman [Put94]).

For problems where the number of states $n$ is very large, simulation-based approaches that are patterned after classical policy iteration methods have been popular (see e.g., [BeT96], [SuB98]). Temporal difference (TD) methods, such as TD($\lambda$) (Sutton [Sut88]), LSPE($\lambda$) (Bertsekas and Ioffe [BeI96]), and LSTD($\lambda$) (Bratdke and Barto [BrB96], Boyan [Boy02]), are commonly used for policy evaluation within this context. The corresponding approximate policy iteration methods have been described in detail in the literature, have been extensively tested in practice, and constitute one of the major methodologies for approximate DP (see the books by Bertsekas and Tsitsiklis [BeT96], Sutton and Barto [SuB98], Gosavi [Gos03], Cao [Cao07], Chang, Fu, Hu, and Marcus [CFH07], Meyn [Mey07], Powell [Pow07], and Borkar [Bor08]; the textbook [Ber07] together with its on-line chapter [Ber10] provide a recent treatment and up-to-date references).

Approximate policy iteration schemes have been used both in a model-based form, and in a model-free form for the computation of Q-factors associated with state-control pairs of given policies. In the latter case, TD methods must contend with a serious difficulty: they generate a sequence of samples $\big\{(i_t, \mu(i_t)), t = 0, 1, \ldots\big\}$ using the Markov chain corresponding to the current policy $\mu$, which means that state-control pairs $(i, u) \neq \big(i, \mu(i)\big)$ are not generated in the simulation. As a result the policy iteration process breaks down as it does not provide meaningful Q-factor estimates for $u \neq \mu(i)$. In practice, it is well-known that it is essential to use an artificial mechanism to ensure that a rich and diverse enough sample of state-control pairs is generated during the simulation.

The use of exploration-enhanced policies is often suggested as a remedy for approximate policy iteration involving TD methods. A common approach, well-known since the early days of approximate DP, is an off-policy strategy (using the terminology of Sutton and Barto [SuB98]; see also Precup, Sutton, and Dasgupta [PSD01]), whereby we occasionally generate transitions involving randomly selected controls rather than the ones dictated by $\mu$. Unfortunately, in the context of Q-learning the required amount of exploration is likely to be substantial, and has an undesirable effect: it may destroy the underlying contraction mapping mechanism on which LSPE($\lambda$) and TD($\lambda$) rely for their validity [see e.g., [BeT96], Example 6.7, which provides an instance of divergence of TD(0)]. At the same time, while LSTD($\lambda$) does not have this difficulty (it does not rely on a contraction property), it requires the solution of a linear projected equation, which has potentially large dimension, particularly when the control constraint sets $U(i)$ have large cardinalities. To address the convergence difficulty in the presence of exploration using an off-policy, the TD($\lambda$) method has been modified in fairly complex ways (Sutton, Szepesvari, and Maei [SSM08], Maei et. al. [MSB08], Sutton et. al. [SMP09]).

The purpose of this paper is to propose an approach to Q-learning with exploration enhancement, which is radically different from existing methods, and is new even in the context of exact DP. It is based on replacing the policy evaluation phase of the classical policy iteration method with (possibly inexact) solution of an *optimal stopping problem*. This problem is defined by a stopping cost and by a *randomized policy*, which are suitably adjusted at the end of each iteration. They encode aspects of the "current policy" and give our algorithm a modified/optimistic policy iteration-like character (a form that is intermediate between value and policy iteration). The randomized policy allows an arbitrary and easily controllable amount of exploration. For extreme choices of the randomized policy and a lookup table representation, our algorithm yields as special cases the classical Q-learning/value iteration and policy iteration methods. Generally, with more exploration and less exact solution of the policy evaluation/optimal stopping problem, the character of the method shifts in the direction of classical Q-learning/value iteration.

We discuss two situations where our algorithm may offer an advantage over existing Q-learning and approximate policy iteration methodology:

(a) In the context of exact/lookup table policy iteration, our algorithm admits asynchronous and stochastic iterative implementations, which can be attractive alternatives to standard methods of asynchronous policy iteration and Q-learning. The advantage of our algorithms is that they involve lower overhead per iteration, by obviating the need for minimization over all controls at every iteration (this is the generic advantage that modified policy iteration has over value iteration).

(b) In the context of approximate policy iteration, with linear Q-factor approximation, our algorithm may be combined with the TD(0)-like method of Tsitsiklis and Van Roy [TsV99], which can be used to solve the associated stopping problems with low overhead per iteration, thereby resolving the issue of exploration described earlier.

Regarding (a) above, note that aside from their conceptual/analytical value, lookup table representation methods can be applied to large scale problems through the use of aggregation (a low-dimensional aggregate representation of a large, possibly infinite-dimensional problem; see Jaakkola, Jordan, and Singh [JJS94], [JSJ95], Gordon [Gor95], Tsitsiklis and Van Roy [TsV96], and Bertsekas [Ber05], [Ber10]). Let us also note that Bhatnagar and Babu [BhB08] have proposed Q-learning/policy iteration type algorithms with lookup table representation, based on two-time-scale stochastic approximation, and established the convergence for synchronous implementations. Their algorithms also have low computation overhead per iteration like our algorithm. However, viewed at the slow-time-scale, their algorithms are close to the standard Q-learning and have a different basis than our algorithm.

The paper is organized as follows. In Section 2, we introduce our policy iteration-like algorithm for the case of exact/lookup table representation of Q-factors, and address convergence issues. In Section 3, we show that our algorithm admits an asynchronous implementation that has improved convergence properties over the standard asynchronous policy iteration algorithm for $Q$-factors. In Section 4, we develop stochastic iterative methods that resemble both Q-learning and modified/optimistic policy iteration, and prove their convergence. In Section 5, we consider the possibility of approximating the policy evaluation portion of our algorithm, and we derive a corresponding error bound, which is consistent with existing error bounds for related methods. In Section 6, we briefly discuss implementations of policy evaluation with linear feature-based approximations and simulation-based optimal stopping algorithms, such as the one due to Tsitsiklis and Van Roy [TsV99]. These algorithms use calculations of low dimension (equal to the number of features), and require low overhead per iteration compared with the matrix inversion overhead required by approximate policy iteration that uses the LSTD($\lambda$) method for policy evaluation.

## 2. A NEW Q-LEARNING ALGORITHM

In this section we introduce our Q-learning algorithm in exact form. We first introduce notation and provide some background. It is well-known that the optimal cost vector $J^*$ is the unique fixed point of the mapping $T : \Re^n \mapsto \Re^n$ given by

$$(TJ)(i) = \min_{u\in U(i)} \sum_{j=1}^n p_{ij}(u)\big(g(i,u,j) + \alpha J(j)\big), \qquad \forall\ i.$$

The optimal Q-factor corresponding to a state-control pair $(i,u)$ is denoted by $Q^*(i,u)$, and represents the optimal expected cost starting from state $x$, using control $u$ at the first stage, and subsequently using an optimal policy. Optimal Q-factors and costs are related by the equation

$$J^*(i) = \min_{u\in U(i)} Q^*(i,u), \qquad \forall\ i. \tag{2.1}$$

The optimal Q-factor vector $Q^*$ is the unique fixed point of the mapping $F$ defined by

$$(FQ)(i,u) = \sum_{j=1}^n p_{ij}(u)\left(g(i,u,j) + \alpha \min_{v\in U(j)} Q(j,v)\right), \qquad \forall\ (i,u). \tag{2.2}$$

One possibility to compute $Q^*$ is the well-known Q-learning algorithm of Watkins [Wat89] (see e.g., [BeT96], [SuB98] for descriptions and discussion), which is an iterative stochastic approximation-like method, based on the fixed point iteration $Q_{k+1} = FQ_k$ for solving the equation $Q = FQ$. Another popular method for computing $Q^*$ is based on policy iteration. At the typical iteration, given the (deterministic stationary) current policy $\mu$, we find $Q_\mu$, the unique fixed point of the mapping $F_\mu$ corresponding to $\mu$, and given by

$$(F_\mu Q)(i,u) = \sum_{j=1}^n p_{ij}(u)\Big(g(i,u,j) + \alpha Q\big(j,\mu(j)\big)\Big), \qquad \forall\ (i,u), \tag{2.3}$$

(this is the policy evaluation step). We then obtain a new policy $\overline{\mu}$ by

$$\overline{\mu}(i) = \arg\min_{u\in U(i)} Q_\mu(i,u), \qquad \forall\ i, \tag{2.4}$$

(this is the policy improvement step).

In this section we propose an alternative policy iteration-like method. The key idea is to replace the Q-learning mapping $F_\mu$ of Eq. (2.3) with another mapping that allows exploration as well as a dependence on $\mu$. This mapping, denoted $F_{J,\nu}$, depends on a vector $J \in \Re^n$, with components denoted $J(i)$, and on a randomized policy $\nu$, which for each state $i$ defines a probability distribution

$$\big\{\nu(u \mid i) \mid u \in U(i)\big\}$$

over the feasible controls at $i$. It maps $Q$, a vector of Q-factors, to $F_{J,\nu}Q$, the vector of Q-factors with components given by

$$(F_{J,\nu}Q)(i,u) = \sum_{j=1}^{n} p_{ij}(u)\left(g(i,u,j) + \alpha \sum_{v\in U(j)} \nu(v \mid j)\min\big\{J(j), Q(j,v)\big\}\right), \qquad \forall\ (i,u). \tag{2.5}$$

Comparing $F_{J,\nu}$ and the classical Q-learning mapping of Eq. (2.2) [or the mapping $F_\mu$ of Eq. (2.3)], we see that they take into account the Q-factors of the next state $j$ differently: $F$ (or $F_\mu$) uses the minimal Q-factor $\min_{v\in U(j)} Q(j,v)$ [the Q-factor $Q\big(j,\mu(j)\big)$, respectively], while $F_{J,\nu}$ uses a randomized Q-factor [according to $\nu(v \mid j)$], but only up to the threshold $J(j)$. Note that $F_{J,\nu}$ does not require the overhead for minimization over all controls that the Q-learning mapping $F$ does [cf. Eq. (2.2)].

The mapping $F_{J,\nu}$ can be interpreted in terms of an optimal stopping problem defined as follows:

(a) The state space is the set of state-control pairs $(i,u)$ of the original problem.

(b) When at state $(i,u)$, if we decide to stop, we incur a stopping cost $J(i)$ (independent of $u$).

(c) When at state $(i,u)$, if we decide not to stop, we incur a one-stage cost $\sum_{j=1}^n p_{ij}(u)g(i,u,j)$, and transition to state $(j,v)$ with probability $p_{ij}(u)\nu(v \mid j)$.

From well-known general properties of Q-learning for MDP, it can be seen that $F_{J,\nu}$ is a sup-norm contraction of modulus $\alpha$ for all $\nu$ and $J$, i.e.,

$$\|F_{J,\nu}Q - F_{J,\nu}\tilde{Q}\|_\infty \le \alpha\|Q - \tilde{Q}\|_\infty, \qquad \forall\ Q,\ \tilde{Q}, \tag{2.6}$$

where $\|\cdot\|_\infty$ denotes the sup-norm ($\|Q\|_\infty = \max_{(i,u)} \big|Q(i,u)\big|$). Hence $F_{J,\nu}$ has a unique fixed point, which we denote by $Q_{J,\nu}$. We may interpret $Q_{J,\nu}(i,u)$ as a Q-factor of the optimal stopping problem corresponding to the nonstopping action, i.e., the optimal cost-to-go starting at $(i,u)$ and conditioned on the first decision being not to stop. Another insight is that if $J$ is the cost of some policy $\pi$, which can be randomized and history dependent, then we may interpret the components of $Q_{J,\nu}$, as the Q-factors of a policy which switches optimally from following the policy $\nu$ to following the policy $\pi$.

For a given $(J,\nu)$, the optimal stopping problem can be solved exactly by using value iteration. When linear feature-based Q-factor approximation is used, it can be solved with the algorithm of Tsitsiklis and Van

Roy [TsV99], a simulation-based TD(0)-type method that uses low-dimensional computation [of order O($s$)] at each iteration and does not require an $s \times s$ matrix inversion (like LSTD or LSPE). Later, in Sections 5 and 6, we will envision the use of this algorithm for approximating $Q_{J,\nu}$.

Note that if $\nu = \mu$, where $\mu$ is a deterministic policy, we have $Q_{J,\mu} \leq Q_\mu$ for all $J$, with equality holding if $J_\mu \leq J$. To get an indication that the mapping $F_{J,\mu}$ can have an advantage in some cases over the Q-learning mapping $F_\mu$, suppose that $J$ is a known upper bound to $J_\mu$ (for example, in the context of policy iteration, $J$ may be the cost vector of the policy preceding $\mu$). Then it can be seen that $Q_\mu \leq F_{J,\mu}Q \leq F_\mu Q$ for all $Q \geq Q_\mu$, which in turn by using induction, shows that

$$Q_\mu \leq F^k_{J,\mu}Q \leq F^k_\mu Q, \qquad \forall\ k = 0, 1, \ldots,$$

i.e., that starting from $Q \geq Q_\mu$, value iteration/Q-learning using $F_{J,\mu}$ converges to $Q_\mu$ at least as fast as it converges using $F_\mu$. Indeed, simple 2-state examples show that the differences between the components of $F^k_{J,\mu}Q$ and $F^k_\mu Q$ can be substantial [take $n = 2$, $g(i,u,j) \equiv 0$, $p_{12}(u) = p_{21}(u) \equiv 1$, $Q(1,u) \equiv J(1) = 1$, $Q(2,u) \equiv J(2) = \beta > 1$]. Therefore, in certain circumstances, iterative evaluation of the Q-factors of a policy $\mu$ may converge substantially faster using $F_{J,\mu}$ than using $F_\mu$. In this paper, however, we focus primarily on other advantages, which are related to asynchronous implementations and exploration, and will be explained in what follows.

The following proposition generalizes the contraction property (2.6). In the proof and for the remainder of the paper, $J^x$ denotes the vector $J$ extended to the space of state-control pairs by

$$J^x(i,u) = J(i), \qquad \forall\ u \in U(i).$$

Furthermore, minimization over two vectors is interpreted componentwise, i.e., $\min\{Q_1, Q_2\}$ denotes the vector with components $\min\big\{Q_1(i,u), Q_2(i,u)\big\}$.

**Proposition 2.1:** For all $\nu$, $J$, $\tilde{J}$, $Q$, and $\tilde{Q}$, we have

$$\|F_{J,\nu}Q - F_{\tilde{J},\nu}\tilde{Q}\|_\infty \leq \alpha \max\big\{\|J - \tilde{J}\|_\infty,\ \|Q - \tilde{Q}\|_\infty\big\}.$$

**Proof:** We write

$$F_{J,\nu}\,Q = \bar{g} + \alpha\overline{P}_\nu \min\big\{J^x, Q\big\}, \tag{2.7}$$

where $\bar{g}$ is the vector with components

$$\sum_{j=1}^n p_{ij}(u)g(i,u,j), \qquad \forall\ (i,u),$$

and $\overline{P}_\nu$ is the transition probability matrix with probabilities of transition $(i,u) \to (j,v)$ equal to

$$p_{ij}(u)\nu(v \mid j), \qquad \forall\ (i,u),\ (j,v).$$

From Eq. (2.7), we obtain

$$\|F_{J,\nu}Q - F_{\tilde{J},\nu}\tilde{Q}\|_\infty \le \alpha \big\| \min\big\{J^x, Q\big\} - \min\big\{\tilde{J}^x, \tilde{Q}\big\}\big\|_\infty.$$

We also have†

$$\big\| \min\big\{J^x, Q\big\} - \min\big\{\tilde{J}^x, \tilde{Q}\big\}\big\|_\infty \le \max\big\{\|J - \tilde{J}\|_\infty,\ \|Q - \tilde{Q}\|_\infty\big\}.$$

The preceding two relations imply the result. **Q.E.D.**

Our Q-learning algorithm generates a sequence of pairs $(Q_k, J_k)$, starting from an arbitrary pair $(Q_0, J_0)$. Given $(Q_k, J_k)$, we select an arbitrary randomized policy $\nu_k$ and an arbitrary positive integer $m_k$, and we obtain the next pair $(Q_{k+1}, J_{k+1})$ as follows:

**Iteration $k$ with Lookup Table Representation:**

(1) Generate $Q_{k+1}$ with $m_k$ iterations involving the mapping $F_{J_k,\nu_k}$, with $\nu_k$ and $J_k$ held fixed:

$$Q_{k+1} = F^{m_k}_{J_k,\nu_k} Q_k. \tag{2.8}$$

(2) Update $J_{k+1}$ by

$$J_{k+1}(i) = \min_{u \in U(i)} Q_{k+1}(i,u), \qquad \forall\ i. \tag{2.9}$$

We will show shortly that $Q_k$ and $J_k$ converge to the optimal Q-factor and cost vector of the original MDP, respectively, but we first discuss the qualitative behavior of the algorithm. To this end, we first

† Here we are using a nonexpasiveness property of the minimization map: for any $Q_1$, $Q_2$, $\tilde{Q}_1$, $\tilde{Q}_2$, we have

$$\big\| \min\{Q_1, Q_2\} - \min\{\tilde{Q}_1, \tilde{Q}_2\}\big\|_\infty \le \max\big\{\|Q_1 - \tilde{Q}_1\|_\infty, \|Q_2 - \tilde{Q}_2\|_\infty\big\}.$$

To see this, write for every $(i,u)$,

$$Q_m(i,u) \le \max\big\{\|Q_1 - \tilde{Q}_1\|_\infty, \|Q_2 - \tilde{Q}_2\|_\infty\big\} + \tilde{Q}_m(i,u), \qquad m = 1,2,$$

take the minimum of both sides over $m$, exchange the roles of $Q_m$ and $\tilde{Q}_m$, and take maximum over $(i,u)$.

consider the two extreme cases where $m_k = 1$ and $m_k = \infty$. For $m_k = 1$,

$$\begin{aligned} Q_{k+1}(i,u) &= \sum_{j=1}^{n} p_{ij}(u) \left( g(i,u,j) + \alpha \sum_{v \in U(j)} \nu_k(v \mid j) \min \left\{ \min_{v' \in U(j)} Q_k(j,v'), Q_k(j,v) \right\} \right) \\ &= \sum_{j=1}^{n} p_{ij}(u) \left( g(i,u,j) + \alpha \min_{v \in U(j)} Q_k(j,v) \right), \qquad \forall\ (i,u), \end{aligned}$$

so Eq. (2.8) coincides with the synchronous Q-learning algorithm $Q_{k+1} = FQ_k$, while Eq. (2.9) coincides with the value iteration $J_{k+1} = TJ_k$ for the original MDP.

On the other hand, in the limiting case where $m_k = \infty$, $Q_{k+1}$ is the Q-factor $Q_{J_k,\nu_k}$ of the associated stopping problem (the unique fixed point of $F_{J_k,\nu_k}$), and the algorithm takes the form

$$J_{k+1}(i) = \min_{u \in U(i)} Q_{J_k,\nu_k}(i,u), \qquad \forall\ i. \tag{2.10}$$

Assume further that $\nu_k$ is chosen to be the deterministic policy $\mu_k$ that attains the minimum in the equation

$$\mu_k(i) = \arg \min_{u \in U(i)} Q_k(i,u), \qquad \forall\ i, \tag{2.11}$$

with $\nu_0$ being some deterministic policy $\mu_0$ satisfying $J_0 \geq J_{\mu_0}$. Then $Q_1$ is equal to $Q_{J_0,\mu_0}$ (since $m_k = \infty$) and can be seen to be also equal to the (exact) Q-factor vector of $\mu_0$ (since $J_0 \geq J_{\mu_0}$), so $\mu_1$ as generated by Eq. (2.11), is the policy generated from $\mu_0$ by exact policy improvement for the original MDP. Similarly, it can be shown by induction that for $m_k = \infty$ and $\nu_k = \mu_k$, the algorithm generates the same sequence of policies as exact policy iteration for the original MDP.

Generally, the iteration (2.8), (2.9) resembles in some ways the classical *modified policy iteration* for MDP (see e.g., [Ber07], [Put94]), where policy evaluation is approximated with a finite number $m_k$ of value iterations, with the case $m_k = 1$ corresponding to value iteration/synchronous Q-learning, and the case $m_k = \infty$ corresponding to (exact) policy iteration.

However, our algorithm has another qualitative dimension, because the randomized policy $\nu_k$ may differ significantly from the deterministic policy (2.11). In particular, suppose that $m_k = \infty$ and $\nu_k$ is chosen to assign positive probability to nonoptimal controls, i.e., so that $\nu_k\big(\mu^*(j) \mid j\big) = 0$ for all $j$ and optimal policies $\mu^*$. Then since $J_k \to J^*$ (as we will show shortly), we have for all $j$ and sufficiently large $k$, $J_k(j) < Q_{J_k,\nu_k}(j,v)$ for all $v$ with $\nu_k(v \mid j) > 0$, so that

$$\begin{aligned} J_{k+1}(i) &= \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \left( g(i,u,j) + \alpha \sum_{v \in U(j)} \nu_k(v \mid j) \min\big\{ J_k(j), Q_{J_k,\nu_k}(j,v) \big\} \right) \\ &= \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \big( g(i,u,j) + \alpha J_k(j) \big), \qquad \forall\ i. \end{aligned}$$

Thus the algorithm, for sufficiently large $k$, reduces to synchronous Q-learning/value iteration for the original MDP, even though $m_k = \infty$, and produces the same results as with the choice $m_k = 1$ (or any value of $m_k$)!

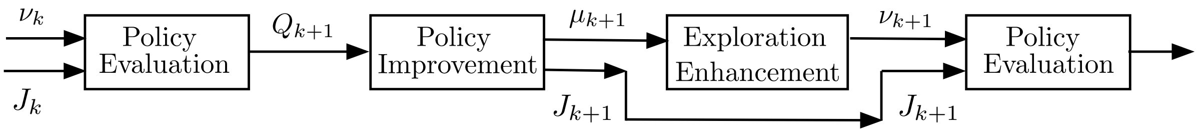

**Figure 2.1.** Illustration of exploration-enhanced policy iteration algorithm. The policy evaluation consists of a finite number of Q-value iterations for the optimal stopping problem involving the randomized policy $\nu$ and the theshold/stopping cost $J$ [cf. Eq. (2.8)]. It is followed by policy improvement that produces a new deterministic policy [cf. Eq. (2.11)], which forms the basis for constructing the new randomized policy using some exploration mechanism.

The preceding arguments illustrate that the choices of $\nu_k$ and $m_k$ are the two factors that affect most the qualitative character of the algorithm. With little exploration [approaching the extreme case where $\nu_k$ is the deterministic policy (2.11)] our algorithm tends to act nearly like modified policy iteration (or exact policy iteration for $m_k = \infty$). With substantial exploration [approaching the extreme case where $\nu_k\big(\mu_k(j) \mid j\big) = 0$ for any policy $\mu_k$ generated according to Eq. (2.11)] it tends to act nearly like Q-learning/value iteration (regardless of the value of $m_k$). This reasoning also suggests that with substantial exploration it may be better to use small values of $m_k$.

When exploration is desired, as in the case where feature-based Q-factor approximations are used (cf. Sections 5 and 6), a reasonable way to operate the algorithm is to determine $\nu_k$ by "superimposing" some exploration to the deterministic policy $\mu_k$ of Eq. (2.11). For example, we may use a distribution $\nu_k$ that is a random mixture of $\mu_k$ and another policy that induces exploration, including visits to state-control pairs that are unlikely/impossible to generate under $\mu_k$). In this case, we may view the calculation of $Q_{k+1}$ via Eq. (2.8) as a form of approximate policy evaluation, somewhat similar to one or more value iterations, depending on the degree of exploration allowed by $\nu_k$ and the value of $m_k$, and we may view Eq. (2.11) as a form of corresponding policy improvement (see Fig. 2.1).

We now prove our main convergence result.

**Proposition 2.2:** For any choice of $(Q_0, J_0)$, $\{\nu_k\}$, and $\{m_k\}$, a sequence $\big\{(Q_k, J_k)\big\}$ generated by the algorithm (2.8)-(2.9) converges to $(Q^*, J^*)$, and the rate of convergence is geometric. Furthermore, for all $k$ after some index $\overline{k}$, the generated policies $\mu_k$ are optimal.

**Proof:** Since $J^*(i) = \min_{u \in U(i)} Q^*(i,u)$ [cf. Eq. (2.1)], we have using Eqs. (2.2) and (2.5), $F_{J^*,\nu}Q^* = FQ^* = Q^*$ for all $\nu$. From Prop. 2.1, it follows that

$$\|F_{J,\nu}Q - Q^*\|_\infty \le \alpha \max\big\{\|J - J^*\|_\infty,\, \|Q - Q^*\|_\infty\big\}, \qquad \forall\ Q,\ J,\ \nu.$$

Using this relation, we have

$$\|F^2_{J_k,\nu_k}Q_k - Q^*\|_\infty \le \alpha \max\big\{\|J_k - J^*\|_\infty,\, \|F_{J_k,\nu_k}Q_k - Q^*\|_\infty\big\} \le \max\big\{\alpha\|J_k - J^*\|_\infty,\, \alpha^2\|Q_k - Q^*\|_\infty\big\},$$

and by repeating this process,

$$\|Q_{k+1} - Q^*\|_\infty = \|F^{m_k}_{J_k,\nu_k}Q_k - Q^*\|_\infty \le \max\big\{\alpha\|J_k - J^*\|_\infty,\, \alpha^{m_k}\|Q_k - Q^*\|_\infty\big\}. \tag{2.12}$$

Since for all $Q$, and $\tilde{Q}$, we have †

$$\max_{i=1,\ldots,n}\left|\min_{u\in U(i)} Q(i,u) - \min_{u\in U(i)} \tilde{Q}(i,u)\right| \le \|Q - \tilde{Q}\|_\infty, \tag{2.13}$$

it follows by taking $Q = Q_k$ and $\tilde{Q} = Q^*$, that for $k > 0$,

$$\|J_k - J^*\|_\infty \le \|Q_k - Q^*\|_\infty. \tag{2.14}$$

Combining Eqs. (2.12) and (2.14), we obtain

$$\|Q_{k+1} - Q^*\|_\infty \le \alpha\|Q_k - Q^*\|_\infty. \tag{2.15}$$

Thus $Q_k$ converges to $Q^*$ geometrically, and in view of Eq. (2.14), $\{J_k\}$ also converges to $J^*$ geometrically. The optimality of $\mu_k$ for sufficiently large $k$ follows from the convergence $Q_k \to Q^*$, since a policy $\mu^*$ is optimal if and only if $\mu^*(i)$ minimizes $Q^*(i,u)$ over $U(i)$ for all $i$. **Q.E.D.**

The preceding proof can also be used to establish a fact that complements Prop. 2.1, namely that for every randomized policy $\nu$ and integer $m \ge 1$, the mapping underlying our algorithm,

$$(Q, J) \quad \mapsto \quad \Big(F^m_{J,\nu}Q,\, M\, F^m_{J,\nu}Q\Big),$$

where

$$(M\, F^m_{J,\nu}Q)(i) = \min_{u\in U(i)} (F^m_{J,\nu}Q)(i,u), \qquad \forall\ i = 1,\ldots,n,$$

is a sup-norm contraction of modulus $\alpha$, and its unique fixed point is $(Q^*, J^*)$. This is the mathematical foundation for the convergence properties of the algorithm (2.8)-(2.9), as well as its asynchronous variants to be discussed in the next section.

† This is a well-known property. For a proof, write

$$Q(i,u) \le \|Q - \tilde{Q}\|_\infty + \tilde{Q}(i,u), \qquad \forall\ (i,u),$$

take minimum of both sides over $u \in U(i)$, exchange the roles of $Q$ and $\tilde{Q}$, and take maximum over $i$.

## 3. ASYNCHRONOUS VERSION OF THE ALGORITHM

The algorithm, as given in Eqs. (2.8)-(2.9), may be viewed as synchronous in the sense that the Q-factors of all state-control pairs are simultaneously updated at each iteration. The contraction property of the underlying mappings [cf. Prop. 2.1 and Eq. (2.13)] can be used to establish the convergence of the algorithm under far more irregular conditions. In particular, we consider in this section asynchronous updating of Q-factors and state costs corresponding to blocks of components, and we discuss in Section 4 model-free sampled versions, which do not require the explicit knowledge of $p_{ij}(u)$ and the calculation of expected values.

In standard asynchronous versions of policy iteration for Q-factors [cf. Eqs. (2.3)-(2.4)], the updates of $\mu$ and $Q$ are executed selectively, for only some of the states and state-control pairs. In a fairly general implementation discussed in the literature ([BeT96], Section 2.2, or [Ber07], Section 1.3.3), there are two types of iterations: those corresponding to an index subset $K_Q$ where $Q$ is updated, and those corresponding to the complementary subset $K_\mu$ where $\mu$ is updated. The algorithm generates a sequence of pairs $(Q_k, \mu_k)$, starting from an arbitrary pair $(Q_0, \mu_0)$ as follows:

$$Q_{k+1}(i,u) = \begin{cases} (F_{\mu_k} Q_k)(i,u) & \text{if } (i,u) \in R_k, \\ Q_k(i,u) & \text{if } (i,u) \notin R_k, \end{cases} \qquad \forall\ k \in K_Q, \tag{3.1}$$

$$\mu_{k+1}(j) = \begin{cases} \arg\min_{v \in U(j)} Q_k(j,v) & \text{if } j \in S_k, \\ \mu_k(j) & \text{if } j \notin S_k, \end{cases} \qquad \forall\ k \in K_\mu, \tag{3.2}$$

where $R_k$ and $S_k$ are subsets of state-control pairs and states, respectively, one of which is nonempty while the other is empty [so that either Eq. (3.1), or Eq. (3.2) is performed]. Relative to ordinary Q-learning, the advantage is that the minimization in Eq. (3.2) is performed only for $k \in K_\mu$ and only for the states in $S_k$ (rather than at each iteration, and for all states), thereby saving computational overhead (this is the generic advantage that modified policy iteration has over ordinary value iteration). Unfortunately, the convergence of the asynchronous policy iteration (3.1)-(3.2) to $Q^*$ is questionable in the absence of additional restrictions; some assumption, such as $F_{\mu_0} Q_0 \leq Q_0$, is required for the initial policy $\mu_0$ and vector $Q_0$ (see [BeT96], Prop. 2.5, or [Ber07], Prop. 1.3.5, and a counterexample by Williams and Baird [WiB93]). The restriction $F_{\mu_0} Q_0 \leq Q_0$ can be satisfied by adding to $Q_0$ a sufficiently large multiple of the unit vector. The need for it, however, indicates that the convergence properties of the algorithm (3.1)-(3.2) are fragile and sensitive to the assumptions, which may cause convergence difficulties in its stochastic simulation-based variants. In particular, no related convergence results or counterexamples are currently known for the case where the expected value of Eq. (3.1) is replaced by a single sample in a stochastic approximation-type of update.

In a corresponding asynchronous version of our algorithm (2.8)-(2.9), again $Q$ is updated selectively, for only some of the state-control pairs, and $J$ is also updated at some iterations and for some of the states.

There may also be a policy $\mu$ that is maintained and updated selectively at some of the states. This policy may be used to generate a randomized policy $\nu$ which enters the algorithm in a material way. However, the algorithm is valid for any choice of $\nu$, so its definition need not involve the policy $\mu$ and the method in which it is used to update $\nu$ (we will later give an example of an updating scheme for $\mu$ and $\nu$). Specifically, our asynchronous algorithm, stated in general terms, generates a sequence of pairs $(Q_k, J_k)$, starting from an arbitrary pair $(Q_0, J_0)$. Given $(Q_k, J_k)$, we obtain the next pair $(Q_{k+1}, J_{k+1})$ as follows:

**Asynchronous Policy Iteration:**

Select a randomized policy $\nu_k$, a subset $R_k$ of state-control pairs, and a subset of states $S_k$ such that $R_k \cup S_k \neq \emptyset$, generate $Q_{k+1}$ according to

$$Q_{k+1}(i,u) = \begin{cases} (F_{J_k,\nu_k} Q_k)(i,u) & \text{if } (i,u) \in R_k, \\ Q_k(i,u) & \text{if } (i,u) \notin R_k, \end{cases} \tag{3.3}$$

and generate $J_{k+1}$ according to

$$J_{k+1}(i) = \begin{cases} \min_{u \in U(i)} Q_k(i,u) & \text{if } i \in S_k, \\ J_k(i) & \text{if } i \notin S_k. \end{cases} \tag{3.4}$$

As mentioned earlier, the preceding algorithm as stated does not have the form of policy iteration. However, $\nu_k$ may be selected in special ways so that it gives the algorithm a policy iteration character, which can then be compared with (synchronous or asynchronous) modified policy iteration for Q-factors, such as the one of Eqs. (3.1)-(3.2). For an example of such an algorithm, assume that a policy $\mu_k$ is also maintained, which defines $\nu_k$ (so $\nu_k$ is the deterministic policy $\mu_k$). The algorithm updates $Q$ according to

$$Q_{k+1}(i,u) = \begin{cases} (F_{J_k,\mu_k} Q_k)(i,u) & \text{if } (i,u) \in R_k, \\ Q_k(i,u) & \text{if } (i,u) \notin R_k, \end{cases} \tag{3.5}$$

and it updates $J$ and $\mu$ according to

$$J_{k+1}(j) = \begin{cases} \min_{v \in U(j)} Q_k(j,v) & \text{if } j \in S_k, \\ J_k(j) & \text{if } j \notin S_k, \end{cases} \qquad \mu_{k+1}(j) = \begin{cases} \arg\min_{v \in U(j)} Q_k(j,v) & \text{if } j \in S_k, \\ \mu_k(j) & \text{if } j \notin S_k, \end{cases} \tag{3.6}$$

where $R_k$ and $S_k$ are subsets of state-control pairs and states.

We may view Eq. (3.5) as a policy evaluation iteration for the state-control pairs in $R_k$, and Eq. (3.6) as a policy improvement iteration only for the states in $S_k$. In comparing the new algorithm (3.5)-(3.6) with the known algorithm (3.1)-(3.2), we see that the essential difference is that Eq. (3.5) involves the use of $J_k$ and the minimization in the right-hand side, while Eq. (3.1) does not. As we will show in the following proposition, this precludes the kind of anomalous behavior that is exhibited in the Williams and Baird counterexample [WiB93] mentioned earlier. Mathematically, the reason for this may be traced to the presence of the cost vector $J$ in Eq. (3.3) and its special case Eq. (3.5), and the sup-norm contraction in the space of $(Q, J)$, which underlies iterations (3.3)-(3.4) and (3.5)-(3.6) (cf. Prop. 2.1).

The following convergence result bears similarity to general convergence results for asynchronous distributed DP and related algorithms involving sup-norm contractions (see [Ber82], [Ber83], and [BeT89], Section 6.2).

**Proposition 3.1:** Assume that each pair $(i, u)$ is included in the set $R_k$ infinitely often, and each state $i$ is included in the set $S_k$ infinitely often. Then any sequence $\big\{(Q_k, J_k)\big\}$ generated by the algorithm (3.3)-(3.4) converges to $(Q^*, J^*)$.

**Proof:** Let $\{k_j\}$ and $\{\hat{k}_j\}$ be sequences of iteration indices such that $k_0 = 0$, $k_j < \hat{k}_j < k_{j+1}$ for $j = 0, 1, \ldots,$ and for all $j$, each $(i, u)$ is included in $\cup_{k=k_j}^{\hat{k}_j-1} R_k$ at least once, while each $i$ is included in $\cup_{k=\hat{k}_j}^{k_{j+1}-1} S_k$ at least once. Thus, between iterations $k_j$ and $\hat{k}_j$, each component of $Q$ is updated at least once, and between iterations $\hat{k}_j$ and $k_{j+1}$, each component of $J$ is updated at least once.

By using Prop. 2.1, we have for all $k$

$$|Q_{k+1}(i,u) - Q^*(i,u)| \leq \alpha \max\big\{\|J_k - J^*\|_\infty,\, \|Q_k - Q^*\|_\infty\big\}, \qquad \forall\ (i,u) \in R_k, \tag{3.7}$$

$$Q_{k+1}(i,u) = Q_k(i,u), \qquad \forall\ (i,u) \notin R_k. \tag{3.8}$$

Also, by using the nonexpansive property of the minimization operation [cf. Eq. (2.13)], we have for all $k$

$$|J_{k+1}(i) - J^*(i)| \leq \|Q_k - Q^*\|_\infty, \qquad \forall\ i \in S_k, \tag{3.9}$$

$$J_{k+1}(i) = J_k(i), \qquad \forall\ i \notin S_k. \tag{3.10}$$

From these relations, it follows that

$$\max\big\{\|J_{k+1} - J^*\|_\infty,\, \|Q_{k+1} - Q^*\|_\infty\big\} \leq \max\big\{\|J_k - J^*\|_\infty,\, \|Q_k - Q^*\|_\infty\big\}, \qquad \forall\ k = 0, 1, \ldots. \tag{3.11}$$

For each $k \in [\hat{k}_j, k_{j+1}]$, we have from Eqs. (3.7), (3.8),

$$|Q_k(i,u) - Q^*(i,u)| \leq \alpha \max\big\{\|J_{\tilde{k}(i,u,k)} - J^*\|_\infty,\, \|Q_{\tilde{k}(i,u,k)} - Q^*\|_\infty\big\}, \qquad \forall\ (i,u), \tag{3.12}$$

where $\tilde{k}(i,u,k)$ is the last iteration index between $k_j$ and $k$ when the component $Q(i,u)$ is updated. Since each component of $Q$ is updated at least once between iterations $k_j$ and $k \in [\hat{k}_j, k_{j+1}]$, using also Eq. (3.11), it follows that

$$\|Q_k - Q^*\|_\infty \leq \alpha \max\big\{\|J_{k_j} - J^*\|_\infty,\, \|Q_{k_j} - Q^*\|_\infty\big\}, \qquad \forall\, j = 0,1,\ldots,\ k \in [\hat{k}_j, k_{j+1}]. \tag{3.13}$$

Since each component of $J$ is updated at least once between iterations $\hat{k}_j$ and $k_{j+1}$, we have from Eqs. (3.9) and (3.10) that

$$|J_{k_{j+1}}(i) - J^*(i)| \leq \|Q_{\tilde{k}(i)} - Q^*\|_\infty, \qquad \forall\, i = 1,\ldots,n,$$

where $\tilde{k}(i)$ is the last iteration index between $\hat{k}_j$ and $k_{j+1}$ when the component $J(i)$ is updated, so from Eq. (3.13), it follows that

$$\|J_{k_{j+1}} - J^*\|_\infty \leq \alpha \max\big\{\|J_{k_j} - J^*\|_\infty,\, \|Q_{k_j} - Q^*\|_\infty\big\}, \qquad \forall\, j = 0,1,\ldots. \tag{3.14}$$

Combining Eqs. (3.13) and (3.14), we obtain

$$\max\big\{\|J_{k_{j+1}} - J^*\|_\infty,\, \|Q_{k_{j+1}} - Q^*\|_\infty\big\} \leq \alpha \max\big\{\|J_{k_j} - J^*\|_\infty,\, \|Q_{k_j} - Q^*\|_\infty\big\}, \qquad \forall\, j = 0,1,\ldots,$$

so $\max\big\{\|J_{k_j} - J^*\|_\infty,\, \|Q_{k_j} - Q^*\|_\infty\big\} \to 0$ as $j \to \infty$, i.e., that $(Q_{k_j}, J_{k_j}) \to (Q^*, J^*)$ as $j \to \infty$. Using also Eq. (3.11), this implies that the entire sequence $\big\{(Q_k, J_k)\big\}$ converges to $(Q^*, J^*)$. **Q.E.D.**

## 4. STOCHASTIC ITERATIVE VERSIONS OF THE ALGORITHM

In this section we consider stochastic iterative versions of our algorithm, which are patterned after the classical Q-learning algorithm of Watkins [Wat89], as well as optimistic and modified policy iteration methods ([BeT96], Section 5.4). We will compare our algorithm with the classical Q-learning algorithm, whereby we generate a sequence of state-control pairs $\big\{(i_k, u_k) \mid k = 0,1,\ldots\big\}$ by any probabilistic mechanism that guarantees that each pair $(i,u)$ appears infinitely often with probability 1, and at each time $k$, we generate a successor state $j_k$ according to the distribution $p_{i_k j}(u_k)$, $j = 1,\ldots,n$, and we update only the Q-factor of $(i_k, u_k)$,

$$Q_{k+1}(i_k,u_k) = \big(1 - \gamma_{(i_k,u_k),k}\big)Q_k(i_k,u_k) + \gamma_{(i_k,u_k),k}\Big(g(i_k,u_k,j_k) + \alpha \min_{v \in U(j)} Q_k(j_k,v)\Big), \tag{4.1}$$

while leaving all other components of $Q_k$ unchanged: $Q_{k+1}(i,u) = Q_k(i,u)$ for all $(i,u) \neq (i_k,u_k)$. The positive stepsizes $\gamma_{(i_k,u_k),k}$ may depend on the current pair $(i_k,u_k)$, and must satisfy assumptions that are standard in stochastic approximation methods (i.e., must diminish to 0 at a suitable rate). There are also distributed asynchronous versions of the algorithm (4.1), where $Q_k(j_k,v)$ may be replaced by $Q_{\tau_{k,v}}(j_k,v)$, where $k - \tau_{k,v}$ may be viewed as a nonnegative integer "delay" that depends on $k$ and $v$, as discussed by Tsitsiklis [Tsi94], and other sources on asynchronous stochastic approximation methods such as [TBA86], [BeT89], [Bor98], [ABB02], and [Bor08].

In what follows in this section, we present three model-free optimistic policy iteration algorithms, which update a cost vector $J$ in addition to the Q-factor vector $Q$, similar to the algorithms of Sections 2 and 3. We focus on a specific order of updates (simultaneous updates of selected components of $J$ and $Q$), but other orders may also be considered. We refer to these algorithms as Algorithms I-III, and we briefly describe them below:

(I) This algorithm resembles the classical Q-learning algorithm (4.1), but requires less overhead per iteration [the minimization over $u \in U(j)$ is replaced by a simpler operation]. It also bears similarity with a known partially optimistic TD(0) algorithm, discussed in Section 5.4 of [BeT96], but has improved convergence properties.

(II) This algorithm parallels the asynchronous policy iteration method (3.3)-(3.4) of the preceding section, but is model-free and uses a single sample per state instead of computing an expected value.

(III) This algorithm generalizes the first two, and allows more complex mechanisms for generating state control pairs, as well as "delayed" components of state costs and Q-factors in its iteration. Among others, the extra generality is helpful in addressing implementations in an asynchronous distributed computing system, and also facilitates the convergence analysis, as we will explain later.

We find it useful to present Algorithms I and II first, since they offer different advantages in different situations, and they help to motivate the more general Algorithm III. We establish the convergence of the latter algorithm using the asynchronous stochastic approximation-type convergence framework of Tsitsiklis [Tsi94]. All the variables involved in the algorithms (states, state-control pairs, costs of states, Q-factors, policies, sets of indexes that determine which components are updated, etc) are to be viewed as random variables defined on a common probability space. Specific technical assumptions about their probabilistic properties will be given at the appropriate points later.

**Some Model-Free Optimistic Policy Iteration Algorithms**

Similar to classical Q-learning, our first algorithm generates a sequence of state-control pairs $\big\{(i_k, u_k) \mid k = 0, 1, \ldots\big\}$, and updates only the Q-factor of $(i_k, u_k)$ at iteration $k$, using a positive stepsize $\gamma_{(i_k,u_k),k}$. It also updates a single component of $J$ if $k \in K_J$, where $K_J$ is an infinite subset of indices (which need not be predetermined, but may depend on algorithmic progress). The algorithm may choose $\nu_k$ arbitrarily for each $k$ and with dependence on $(i_k, u_k)$, but one possibility is to maintain a policy $\mu_k$ that is updated at selected states simultaneously with $J$, and then use $\nu_k = \mu_k$, similar to algorithm (3.5)-(3.6). Furthermore, the controls $u_k$ may be generated in accordance with $\nu_k$; this gives the algorithm a modified/optimistic policy iteration character. The states $i_{k+1}$ may be generated according to $p_{i_kj}(u_k)$, as in some optimistic policy iteration methods, although this is not essential for the convergence of the algorithm. Compared to the preceding Q-learning algorithm (4.1), the algorithm has an advantage similar to the one that modified policy iteration has over value iteration [less overhead because it does not require the minimization over all controls $v \in U(j)$ at every iteration]. In particular, given the pair $(Q_k, J_k)$, the algorithm obtains $(Q_{k+1}, J_{k+1})$ as follows:

**Model-Free Optimistic Policy Iteration I:**

(1) Select a state-action pair $(i_k, u_k)$. If $k \in K_J$, update $J_k$ according to

$$J_{k+1}(j) = \begin{cases} \min_{v\in U(j)} Q_k(j,v) & \text{if } j = i_k, \\ J_k(j) & \text{if } j \neq i_k; \end{cases} \tag{4.2}$$

otherwise leave $J_k$ unchanged $(J_{k+1} = J_k)$.

(2) Select a stepsize $\gamma_{(i_k,u_k),k} \in (0,1]$ and a policy $\nu_{(i_k,u_k),k}$. Generate a successor state $j_k$ according to the distribution $p_{i_kj}(u_k)$, $j = 1, \ldots, n$, and generate a control $v_k$ according to the distribution $\nu_{(i_k,u_k),k}(v \mid j_k)$, $v \in U(j_k)$.

(3) Update the $(i_k, u_k)$th component of $Q$ according to

$$Q_{k+1}(i_k, u_k) = \big(1 - \gamma_{(i_k,u_k),k}\big)Q_k(i_k, u_k) + \gamma_{(i_k,u_k),k}\Big(g(i_k, u_k, j_k) + \alpha \min\big\{J_k(j_k), Q_k(j_k, v_k)\big\}\Big), \tag{4.3}$$

and leave all other components of $Q_k$ unchanged: $Q_{k+1}(i,u) = Q_k(i,u)$ for all $(i,u) \neq (i_k, u_k)$.

The preceding algorithm (Algorithm I) has similarities with the partially optimistic TD(0) algorithm, discussed in Section 5.4 of [BeT96]. The latter algorithm updates only $J$ [rather than $(J,Q)$] using TD(0), and also maintains a policy, which is updated at selected iterations. However, its convergence properties are dubious, as discussed in p. 231 of [BeT96] (see also Tsitsiklis [Tsi02]). By contrast, we will show that our algorithm above has satisfactory convergence properties.

We now give another stochastic iterative algorithm, which parallels the asynchronous policy iteration method (3.3)-(3.4) of Section 3. Given the pair $(Q_k, J_k)$, the algorithm obtains $(Q_{k+1}, J_{k+1})$ as follows:

**Model-Free Optimistic Policy Iteration II:**

Select a subset $R_k$ of state-control pairs, and a subset of states $S_k$ such that $R_k \cup S_k \neq \emptyset$.

Update $J_k$ according to

$$J_{k+1}(i) = \begin{cases} \min_{u \in U(i)} Q_k(i,u) & \text{if } i \in S_k, \\ J_k(i) & \text{if } i \notin S_k. \end{cases} \tag{4.4}$$

For each $\ell = (i,u) \in R_k$, select a stepsize $\gamma_{\ell,k} \in (0,1]$ and a policy $\nu_{\ell,k}$, and:

(1) Generate a successor state $j_k$ according to the distribution $p_{ij}(u)$, $j = 1, \ldots, n$, and generate a control $v_k$ according to the distribution $\nu_{\ell,k}(v \mid j_k)$, $v \in U(j_k)$.

(2) Update the $(i,u)$th component of $Q_k$ according to

$$Q_{k+1}(i,u) = \big(1 - \gamma_{(i,u),k}\big) Q_k(i,u) + \gamma_{\ell,k}\Big(g(i,u,j_k) + \alpha \min\big\{J_k(j_k), Q_k(j_k,v_k)\big\}\Big). \tag{4.5}$$

Leave all other components of $Q_k$ unchanged: $Q_{k+1}(i,u) = Q_k(i,u)$ for all $(i,u) \notin R_k$.

In the preceding algorithm (Algorithm II), the successor state-control pair $(j_k, v_k)$ corresponding to the different pairs $\ell = (i,u) \in R_k$ are different random variables. We have used the same notation for simplicity. Compared with Algorithm I, the chief difference in Algorithm II is that it allows multiple components of $J$ and $Q$ to be updated at each iteration. Compared with the deterministic asynchronous version (3.3)-(3.4), the chief difference is that selected components of $Q$ are updated using a single sample in place of the expected value that defines $F_{J_k,\nu_{\ell,k}}$ [cf. Eqs. (3.3) and (3.5)]. Such updates must satisfy certain properties, to be discussed in what follows, so that the error due to simulation noise will vanish in the limit.

It is convenient to view the next algorithm (Algorithm III) as an algorithm that operates in the joint space of the pair $(J,Q)$. We denote $x_k = (J_k, Q_k)$ and introduce outdated information in updating $x_k$. This is natural for asynchronous distributed computation, in which case each component $\ell$ may be associated with a processor, which keeps at time $k$ a local, outdated version of $x_k$, denoted by $x_k^{(\ell)}$. We introduce outdated information not just for more generality, but also to facilitate the association with the algorithmic framework of [Tsi04], which we will use in our convergence proof. In particular, $x_k^{(\ell)}$ has the form

$$x_k^{(\ell)} = \big(x_{1,\tau_{1,k}^{\ell}}, \ldots, x_{m,\tau_{m,k}^{\ell}}\big), \tag{4.6}$$

where the nonnegative difference $k - \tau_{j,k}^{\ell}$ indicates a “communication delay” relative to the “current” time $k$ for the $j$th component of $x$ at the processor updating component $\ell$ ($j, \ell = 1, \ldots, m$, with $m$ being the sum of the number of states and the number of state-control pairs). We write $x_k^{(\ell)}$ in terms of its components $J$ and $Q$ as

$$x_k^{(\ell)} = \big(J_k^{(\ell)}, Q_k^{(\ell)}\big). \tag{4.7}$$

We will require later that $\lim_{k\to\infty} \tau_{j,k}^\ell = \infty$ for all $\ell$ and $j$, but the exact values of $\tau_{j,k}^\ell$ are immaterial and need not even be known to the processor.

In the following Algorithm III, we can use outdated information to update $J$ and $Q$, and the choice of the policy $\nu$ at time $k$ may depend on the successor state $j_k$ in addition to the history of the algorithm up to time $k$. To be more precise, let $I_k$ be an information vector, a random variable that consists of the entire history of the algorithm up to time $k$ (this includes the stepsizes $\gamma_{\ell,t}$, the index sets $S_t$ and $R_t$ selected for cost and Q-factor updates, the results of the updates, and the delays $t - \tau_{j,t}^\ell$, at all times $t \le k$). We will assume that the selection of the policy is based on $(I_k, j_k)$, where $j_k$ is the successor state generated according to probabilities $p_{ij}(u)$ similar to Algorithm II.

**Model-Free Optimistic Policy Iteration III:**

Select a subset $R_k$ of state-control pairs, and a subset of states $S_k$ such that $R_k \cup S_k \ne \emptyset$. For each $\ell \in R_k \cup S_k$, choose a stepsize $\gamma_{\ell,k} \in (0,1]$ and times $\tau_{j,k}^\ell \le k$, $j = 1, \ldots, m$. Let $(J_k^{(\ell)}, Q_k^{(\ell)})$ be as defined in Eqs. (4.6) and (4.7).

Update $J_k$ according to

$$J_{k+1}(i) = \begin{cases} (1-\gamma_{\ell,k})J_k(i) + \gamma_{\ell,k}\min_{u\in U(i)} Q_k^{(\ell)}(i,u), \text{ with } \ell = i, & \text{if } i \in S_k, \\ J_k(i) & \text{if } i \notin S_k. \end{cases} \tag{4.8}$$

For each $\ell = (i,u) \in R_k$:

(1) Generate a successor state $j_{\ell,k}$ according to the distribution $p_{ij}(u)$, $j = 1, \ldots, n$. Select a policy $\nu_{\ell,I_k,j_{\ell,k}}$ based on the information $(I_k, j_{\ell,k})$, and generate a control $v_{\ell,k}$ according to the distribution $\nu_{\ell,I_k,j_{\ell,k}}(v \mid j_{\ell,k})$, $v \in U(j_{\ell,k})$.

(2) Update the $(i,u)$th component of $Q_k$ according to

$$Q_{k+1}(i,u) = \big(1-\gamma_{\ell,k}\big)Q_k(i,u) + \gamma_{\ell,k}\Big(g(i,u,j_{\ell,k}) + \alpha \min\big\{J_k^{(\ell)}(j_{\ell,k}), Q_k^{(\ell)}(j_{\ell,k}, v_{\ell,k})\big\}\Big). \tag{4.9}$$

Leave all other components of $Q_k$ unchanged: $Q_{k+1}(i,u) = Q_k(i,u)$ for all $(i,u) \notin R_k$.

### A General Algorithmic Model

As preparation for an analytically more convenient description of Algorithm III, we introduce some notation. Let $\mathcal{M}$ denote the set of all stationary (deterministic or randomized) policies. For each $\nu \in \mathcal{M}$, define an operator $L^\nu$ on the space of $(J, Q)$ by

$$(\tilde{J}, \tilde{Q}) = L^\nu(J, Q), \tag{4.10}$$

where

$$\tilde{J}(i) = \min_{u \in U(i)} Q(i, u), \quad i = 1, \ldots, n, \qquad \tilde{Q} = F_{J,\nu} Q. \tag{4.11}$$

Denote the $\ell$th component of the mapping $L^\nu$ by $L^\nu_\ell$, where $\ell = 1, \ldots, m$. As can be seen from Eq. (4.11), if $\ell$ corresponds to the $i$th component of $J$, then $L^\nu_\ell(J, Q) = \min_{u \in U(i)} Q(i, u)$, whereas if $\ell$ corresponds to the $(i, u)$th component of $Q$, then $L^\nu_\ell(J, Q) = (F_{J,\nu} Q)(i, u)$.

We note that for a given $\ell \in R_k$, the policy $\nu_{\ell, I_k, j_{\ell,k}}$ is a measurable $\mathcal{M}$-valued random variable with respect to the $\sigma$-field $\sigma(I_k, j_{\ell,k})$ generated by $(I_k, j_{\ell,k})$ [since it is selected with knowledge of $(I_k, j_{\ell,k})$]. We introduce the $\sigma(I_k)$-measurable $\mathcal{M}$-valued random variable $\bar{\nu}_{\ell, I_k} = \big\{\bar{\nu}_{\ell, I_k}(v \mid j) \mid v \in U(j), j = 1, \ldots, n\big\}$, which is the conditional distribution of $v$ corresponding to the joint distribution $P\,(j_{\ell,k} = j,\, v_{\ell,k} = v \mid I_k)$, i.e.,

$$P\,(j_{\ell,k} = j,\, v_{\ell,k} = v \mid I_k) = p_{ij}(u)\, \bar{\nu}_{\ell, I_k}(v \mid j), \qquad \forall\, j,\ v \in U(j). \tag{4.12}$$

[If $\ell = (i, u)$ and $j$ is such that $p_{ij}(u) = 0$, we have $P\,(j_{\ell,k} = j,\, v_{\ell,k} = v \mid I_k) = 0$ for all $v \in U(j)$, and we may define $\bar{\nu}_{\ell, I_k}(v \mid j)$ to be any distribution over $U(j)$, for example the uniform distribution.] Note that if in Algorithm III, $\nu_{\ell, I_k, j_{\ell,k}}(\cdot \mid j)$ is chosen before $j_{\ell,k}$ is generated, then $\bar{\nu}_{\ell, I_k}$ coincides with $\nu_{\ell,k}$; this is the case in Algorithm II.

We can now express Algorithm III in a compact form using the mappings $L^\nu$ of Eqs. (4.10)-(4.11). It can be equivalently written as

$$x_{\ell,k+1} = (1 - \gamma_{\ell,k}) x_{\ell,k} + \gamma_{\ell,k} \Big( L^{\bar{\nu}_{\ell, I_k}}_\ell \big(x^{(\ell)}_k\big) + w_{\ell,k} \Big), \tag{4.13}$$

where:

(a) If $\ell = (i, u) \in R_k$, we have $\gamma_{\ell,k} \in (0, 1]$, and $w_{\ell,k}$ is a noise term given by

$$w_{\ell,k} = g(i, u, j_{\ell,k}) + \alpha \min \big\{ J^{(\ell)}_k(j_{\ell,k})\,,\, Q^{(\ell)}_k(j_{\ell,k}, v_{\ell,k}) \big\} - \Big( F_{J^{(\ell)}_k, \bar{\nu}_{\ell, I_k}} Q^{(\ell)}_k \Big)(i, u). \tag{4.14}$$

[cf. Eqs. (4.9) and (4.11), and noticing that $L^{\bar{\nu}_{\ell, I_k}}_\ell (x^{(\ell)}_k) = (F_{J^{(\ell)}_k, \bar{\nu}_{\ell, I_k}} Q^{(\ell)}_k)(i, u)$.]

(b) If $\ell \in S_k$, we have $\gamma_{\ell,k} \in (0, 1]$, $w_{\ell,k} = 0$, and $\bar{\nu}_{\ell, I_k}$ is immaterial [cf. Eqs. (4.8) and (4.11)].

(c) If $\ell \notin R_k \cup S_k$, we have $\gamma_{\ell,k} = 0$, $w_{\ell,k} = 0$.

With $\gamma_{\ell,k}$ defined for all $\ell$ and $k$, the sets $R_k, S_k$ may also be specified implicitly by those $\gamma_{\ell,k}$ that are positive.

### Convergence Analysis

Our convergence analysis of the general algorithm (4.8)-(4.9), equivalently given in (4.13)-(4.14), uses extensions of two results from Tsitsiklis [Tsi94], which relate to the convergence of algorithms of the form (4.13) with the exception that there is only a single contraction mapping $L$ in place of $L^{\bar{\nu}_{\ell},I_k}$. Our analysis is based on the observation that these results of [Tsi94] extend to the case with multiple mappings, if the latter are contraction mappings with respect to the same norm and have the same fixed point.

Thus, the first step of our convergence proof is to establish a common contraction property of $L^{\nu}$ for all stationary policies $\nu$. Define a weighted sup-norm $\|\cdot\|_{\zeta}$ on the space of $(J,Q)$ by

$$\|(J,Q)\|_{\zeta} = \max\left\{\frac{\|J\|_{\infty}}{\xi}, \|Q\|_{\infty}\right\}, \tag{4.15}$$

where $\xi$ is a positive scalar such that

$$\xi > 1, \quad \alpha\xi < 1. \tag{4.16}$$

**Proposition 4.1:** Let $\|\cdot\|_{\zeta}$ and $\xi$ be given by Eqs. (4.15) and (4.16), respectively, and let $\beta = \max\{\alpha\xi, 1/\xi\} < 1$. For all stationary policies $\nu$, $(J^*,Q^*)$ is the unique fixed point of the mapping $L^{\nu}$ given by Eqs. (4.10)-(4.11), and we have

$$\|L^{\nu}(J,Q) - L^{\nu}(J',Q')\|_{\zeta} \leq \beta\|(J,Q) - (J',Q')\|_{\zeta} \tag{4.17}$$

for all pairs $(J,Q)$ and $(J',Q')$.

**Proof:** At the beginning of the proof of Prop. 2.2 we showed that $(J^*,Q^*)$ is a fixed point of $L^{\nu}$ for all $\nu$. The uniqueness of the fixed point will be implied by Eq. (4.17), which we now prove. Let $(\tilde{J},\tilde{Q}) = L^{\nu}(J,Q)$ and $(\tilde{J}',\tilde{Q}') = L^{\nu}(J',Q')$. By Prop. 2.1, we have

$$\begin{aligned}\|\tilde{Q} - \tilde{Q}'\|_{\infty} &\leq \alpha\max\{\|J-J'\|_{\infty}\,,\,\|Q-Q'\|_{\infty}\}\\ &= \alpha\max\left\{\xi\cdot\frac{\|J-J'\|_{\infty}}{\xi}\,,\;\|Q-Q'\|_{\infty}\right\}\\ &\leq \alpha\max\left\{\xi\cdot\frac{\|J-J'\|_{\infty}}{\xi}\,,\;\xi\cdot\|Q-Q'\|_{\infty}\right\}\\ &= \alpha\xi\cdot\|(J,Q)-(J',Q')\|_{\zeta},\end{aligned} \tag{4.18}$$

where we used $\xi > 1$ to derive the second inequality. We also have

$$\|\tilde{J} - \tilde{J}'\|_\infty \leq \|Q - Q'\|_\infty,$$

which implies that

$$\begin{aligned} \frac{\|\tilde{J} - \tilde{J}'\|_\infty}{\xi} &\leq \frac{1}{\xi} \cdot \|Q - Q'\|_\infty \\ &\leq \frac{1}{\xi} \cdot \max\left\{ \frac{\|J - J'\|_\infty}{\xi}, \|Q - Q'\|_\infty \right\} \\ &= \frac{1}{\xi} \cdot \|(J, Q) - (J', Q')\|_\zeta. \end{aligned} \tag{4.19}$$

Equations (4.18) and (4.19) imply the desired property (4.17):

$$\begin{aligned} \|(\tilde{J}, \tilde{Q}) - (\tilde{J}', \tilde{Q}')\|_\zeta &= \max\left\{ \frac{\|\tilde{J} - \tilde{J}'\|_\infty}{\xi}, \|\tilde{Q} - \tilde{Q}'\|_\infty \right\} \\ &\leq \max\{\alpha\xi, 1/\xi\} \cdot \|(J, Q) - (J', Q')\|_\zeta \\ &= \beta \, \|(J, Q) - (J', Q')\|_\zeta. \end{aligned}$$

**Q.E.D.**

We now specify conditions on the variables involved in the algorithm (4.13)-(4.14). Our conditions parallel the assumptions given in [Tsi94] (Assumptions 1-3), which are standard for asynchronous stochastic approximation. We use the shorthand "w.p.1" for "with probability 1." The first condition is a mild, natural requirement for the delays.

**Condition 4.1:** For any $\ell$ and $j$, $\lim_{k\to\infty} \tau^\ell_{j,k} = \infty$ w.p.1.

The next condition is mainly about the noise terms $w_{\ell,k}$. Let $(\Omega, \mathcal{F}, P)$ be the common probability space on which all the random variables involved in the algorithm are defined, and let $\{\mathcal{F}_k, k \geq 0\}$ be an increasing sequence of subfields of $\mathcal{F}$.

**Condition 4.2:**

(a) $x_0$ is $\mathcal{F}_0$-measurable.

(b) For every $\ell$ corresponding to a component of $Q$ and every $k$, $w_{\ell,k}$ is $\mathcal{F}_{k+1}$-measurable.

(c) For every $j$, $\ell$, and $k$, $\gamma_{\ell,k}$, $\tau^\ell_{j,k}$ and $\bar{\nu}_{\ell,I_k}$ are $\mathcal{F}_k$-measurable.

(d) For every $\ell$ corresponding to a component of $Q$ and every $k$,

$$E\big[w_{\ell,k} \mid \mathcal{F}_k\big] = 0.$$

(e) There exist (deterministic) constants $A$ and $B$ such that for every $\ell$ corresponding to a component of $Q$ and every $k$,

$$E\big[w^2_{\ell,k} \mid \mathcal{F}_k\big] \leq A + B \max_j \max_{\tau \leq k} |x_{j,\tau}|^2.$$

The next condition deals with the stepsize variables.

**Condition 4.3:**

(a) For every $\ell$,

$$\sum_{k\geq 0} \gamma_{\ell,k} = \infty, \quad w.p.1.$$

(b) There exists some (deterministic) constant $C$ such that for every $\ell$ corresponding to a component of $Q$,

$$\sum_{k\geq 0} \gamma_{\ell,k}^2 \leq C, \quad w.p.1.$$

Condition 4.3(a) implies that all components of $J$ and $Q$ are updated infinitely often, which is also part of the assumptions of Prop. 3.1. A simple way to choose stepsize sequences $\{\gamma_{\ell,k}\}$ that satisfy Condition 4.3 is to define them using a positive scalar sequence $\{\gamma_k\}$ which diminishes to 0 at a suitable rate [e.g., $O(1/k)$]: For all $\ell \in R_k$, let $\gamma_{\ell,k}$ have a common value $\gamma_k$, and select all state-control pairs $(i,u)$ "comparably often" in the sense that the fraction of times $(i,u)$ is selected for iteration is nonzero in the limit (see Borkar [Bor08]).

There are two insignificant differences between the preceding conditions and the assumptions in [Tsi94] (Assumptions 1-3). First, Condition 4.2(c) is imposed on the random variables $\bar{\nu}_{\ell,I_k}$, which do not appear in [Tsi94]. Second, Conditions 4.2(d) and 4.2(e) are imposed on the noise terms $w_{\ell,k}$, which are involved in the updates of components of $Q$ only [for components of $J$, there is no noise ($w_{\ell,k}=0$) in the updates and these conditions are trivially satisfied]. For the same reason, Condition 4.3(b), a standard condition for bounding asymptotically the error due to noise, is also imposed on the components of $Q$ only [in [Tsi94], Condition 4.3(b) is imposed on all components of $x$].

We now verify that by its definition, the algorithm (4.13)-(4.14) satisfies Condition 4.2. Let $\mathcal{F}_k = \sigma(I_k)$. Then Conditions 4.2(a)-(c) are satisfied by the definition of the algorithm; in particular, note that $\bar{\nu}_{\ell,I_k}$ is by definition $\mathcal{F}_k$-measurable [cf. Eq. (4.12)]. We verify Conditions 4.2(d)-(e), similar to the standard Q-learning case given in [Tsi94]. Let $\ell \in R_k$ and $(j_{\ell,k}, v_{\ell,k})$ be the corresponding successor state-control pair. From the way $j_{\ell,k}$ is generated, it is seen that

$$E\big[g(i,u,j_{\ell,k}) \mid \mathcal{F}_k\big] = \sum_{j=1}^{n} p_{ij}(u) g(i,u,j).$$

From the way $\big(j_{\ell,k}, v_{\ell,k}\big)$ is generated and the definition of $\bar{\nu}_{\ell,I_k}$ [cf. Eq. (4.12)], we have

$$E\left[\min\left\{J_k^{(\ell)}(j_{\ell,k}), Q_k^{(\ell)}(j_{\ell,k}, v_{\ell,k})\right\} \mid \mathcal{F}_k\right] = \sum_{j=1}^{n} p_{ij}(u) \sum_{v\in U(j)} \bar{\nu}_{\ell,I_k}(v \mid j) \min\big\{J_k^{(\ell)}(j)\,,\, Q_k^{(\ell)}(j,v)\big\}.$$

Taking conditional expectation in Eq. (4.14) and using the preceding two equations, we obtain

$$\begin{aligned} E\big[w_{\ell,k} \mid \mathcal{F}_k\big] &= \sum_{j=1}^{n} p_{ij}(u) \left( g(i,u,j) + \alpha \sum_{v \in U(j)} \bar{\nu}_{\ell,I_k}(v \mid j) \min\big\{J_k^{(\ell)}(j)\,,\, Q_k^{(\ell)}(j,v)\big\} \right) \\ &\quad - \big(F_{J_k^{(\ell)},\bar{\nu}_{\ell,I_k}} Q_k^{(\ell)}\big)(i,u) \\ &= 0, \end{aligned}$$

so Condition 4.2(d) is satisfied. It can also be seen that we may write $w_{\ell,k} = Z_1 + Z_2$ with

$$\begin{aligned} Z_1 &= g(i,u,j_{\ell,k}) - E\big[g(i,u,j_{\ell,k}) \mid \mathcal{F}_k\big], \\ Z_2 &= \alpha \min\Big\{J_k^{(\ell)}(j_{\ell,k})\,,\, Q_k^{(\ell)}\,(j_{\ell,k}, v_{\ell,k})\Big\} - E\left[\alpha \min\Big\{J_k^{(\ell)}(j_{\ell,k})\,,\, Q_k^{(\ell)}\,(j_{\ell,k}, v_{\ell,k})\Big\} \;\Big|\; \mathcal{F}_k\right], \end{aligned}$$

where the first expectation is over $j_{\ell,k}$ and the second is over $(j_{\ell,k}, v_{\ell,k})$. Since the number of state-control pairs is finite, the variance of $g(i,u,j_{\ell,k})$ can be bounded by a constant $C$ for all $(i,u)$: $E\big[Z_1^2 \mid \mathcal{F}_k\big] \leq C$. † The conditional variance of $\min\Big\{J_k^{(\ell)}(j_{\ell,k}), Q_k^{(\ell)}\,(j_{\ell,k}, v_{\ell,k})\Big\}$, conditioned on $\mathcal{F}_k$, is bounded by the square of the largest absolute value that this random variable can possibly take, so

$$E\big[Z_2^2 \mid \mathcal{F}_k\big] \leq \alpha^2 \max_j \max_{\tau \leq k} |x_{j,\tau}|^2.$$

Thus, using also the Cauchy-Schwarz inequality, we have

$$\begin{aligned} E\big[w_{\ell,k}^2 \mid \mathcal{F}_k\big] &\leq C + \alpha^2 \max_j \max_{\tau \leq k} |x_{j,\tau}|^2 + 2\sqrt{C \cdot \alpha^2 \max_j \max_{\tau \leq k} |x_{j,\tau}|^2} \\ &\leq A + B \max_j \max_{\tau \leq k} |x_{j,\tau}|^2, \qquad\qquad \forall\ k,\ \ell \in R_k, \end{aligned}$$

for some deterministic constants $A$ and $B$, so Condition 4.2(e) is satisfied.

**Proposition 4.2:** Under Conditions 4.1 and 4.3, any sequence $\{x_k\}$ with $x_k = (J_k, Q_k)$ generated by the model-free optimistic policy iteration algorithm (4.13)-(4.14) [or equivalently, (4.8)-(4.9)] converges to $x^* = (J^*, Q^*)$ with probability 1.

**Proof:** We have shown that Condition 4.2 is satisfied by the algorithm (4.13)-(4.14), so under the assumption of the proposition, we have that all Conditions 4.1-4.3 hold. We apply the analysis of [Tsi94], and in particular, the proofs of Theorems 1 and 3 of that reference. The two theorems imply the boundedness of $\{x_k\}$ and the convergence of $\{x_k\}$ to $x^*$ with probability 1, respectively, for iterates of the form

$$x_{\ell,k+1} = (1 - \gamma_{\ell,k})x_{\ell,k} + \gamma_{\ell,k}\big(L_\ell\big(x_k^{(\ell)}\big) + w_{\ell,k}\big),$$

† If instead of a scalar, $g(i,u,j)$ is also treated as random, then one may impose a finite variance condition on it.

where $L$ is a contraction mapping with fixed point $x^*$, under assumptions that parallel Conditions 4.1-4.3 with minor differences, which we address below [in our algorithm, there are multiple contraction mappings $L^\nu$ that share the same fixed point, and the condition 4.3(b) is satisfied only for $\ell$ corresponding to components of $Q$].

First, for a contraction mapping $L$ with modulus $\beta$ and with respect to a weighted sup-norm $\|\cdot\|_\zeta$, $L$ enters in the proofs of Theorems 1 and 3 of [Tsi94], only via the two inequalities:

$$\|L(x)\|_\zeta \leq \beta\|x\|_\zeta + D, \quad \forall x, \tag{4.20}$$

where $D$ is some constant, and

$$\|L(x) - x^*\|_\zeta \leq \beta\|x - x^*\|_\zeta, \quad \forall x. \tag{4.21}$$

Implications of these inequalities are used to bound $L_\ell(x_k^{(\ell)})$ in the iterates $x_{\ell,k+1}$ for each sample path from a set of probability one.

Second, in the proofs of Theorems 1 and 3 of [Tsi94], the effect of the noise $\{w_{\ell,k}\}$ on $\{x_{\ell,k}\}$ for each component $\ell$ is analyzed in two lemmas, Lemmas 1 and 2, under Conditions 4.2(b)-(e) and 4.3 for that particular component. It is only in those two places that Condition 4.3(b) for a component is used. The rest of the analysis for Theorems 1 and 3 relies only indirectly on Condition 4.3(b) through the two lemmas.

In our case, the inequalities (4.20) and (4.21) are satisfied by all $L^{\bar{\nu}_{\ell,I_k}}$ for the same $\|\cdot\|_\zeta, \beta, D$, and $x^* = (J^*, Q^*)$, as established in Prop. 4.1. Moreover, when $\ell$ corresponds to a component of $J$, while the stepsizes $\gamma_{\ell,k}$ are not restricted by Condition 4.3(b), because the noise terms $w_{\ell,k}, k \geq 0$ are always zero, Lemmas 1 and 2 of [Tsi94] trivially hold without Condition 4.3(b) for such $\ell$. It then follows that Lemmas 1 and 2 hold for all components $\ell$ of $x$ in our case. We can thus apply the proofs of the two theorems of [Tsi94] with $L_\ell\big(x_k^{(\ell)}\big)$ replaced by $L_\ell^{\bar{\nu}_{\ell,I_k}}\big(x_k^{(\ell)}\big)$ to establish the convergence to $x^*$ with probability 1 for the sequence $\{x_k\}$ generated by the algorithm (4.13)-(4.14). **Q.E.D.**

## 5. ERROR BOUNDS FOR APPROXIMATE IMPLEMENTATIONS

In this section, we discuss the effect of approximations on the algorithm of Section 2. In particular, we consider performing the iteration $Q_{k+1} = F_{J_k,\nu_k}^{m_k} Q_k$ [cf. Eq. (2.8)] approximately, possibly using simulation and function approximation. In such an algorithm, we generate a sequence $\{Q_k\}$ such that

$$\big\|Q_{k+1} - F_{J_k,\nu_k}^{m_k} Q_k\big\|_\infty \leq \delta, \tag{5.1}$$

for some $\delta > 0$ and a sequence of positive integers $\{m_k\}$. We then update $J_k$ according to

$$J_{k+1}(i) = \min_{u \in U(i)} Q_{k+1}(i,u), \qquad \forall\, i, \tag{5.2}$$

and let the randomized policy $\nu_{k+1}$ be arbitrary as before.

The analysis also holds when $m_k$ may be equal to $\infty$, in which case Eq. (5.1) is replaced by

$$\|Q_{k+1} - Q_{J_k,\nu_k}\|_\infty \leq \delta.$$

The computation of $Q_{k+1}$ can be done in a number of ways, some of which are discussed in the next section. In this section, we derive an error bound in the following proposition.

**Proposition 5.1:** Assume that for some $\delta \geq 0$ and each $k \geq 0$, there exists a positive integer $m_k$ such that Eq. (5.1) holds. Let $\mu_{k+1}$ be a policy such that $\mu_{k+1}(i)$ attains the minimum in Eq. (5.2) for all $i$. Then, for any stationary policy $\mu$ that is a limit point of $\{\mu_k\}$, we have

$$\|J_\mu - J^*\|_\infty \leq \frac{2\delta}{(1-\alpha)^2}. \tag{5.3}$$

The bound (5.3) is identical to what is generally viewed as the standard bound for the performance of approximate policy iteration ([BeT96], Prop. 6.2). We prove this bound through three lemmas.

**Lemma 5.1:** For all $\nu$, $J$, $\tilde{J}$, $Q$, $\tilde{Q}$, and $m \geq 1$, we have

$$\big\|F^m_{J,\nu} Q - F^m_{\tilde{J},\nu} \tilde{Q}\big\|_\infty \leq \alpha \max\big\{\|J - \tilde{J}\|_\infty,\, \|Q - \tilde{Q}\|_\infty\big\}.$$

**Proof:** Follows by repeated application of Prop. 2.1. **Q.E.D.**

**Lemma 5.2:** Given $(J,\nu)$ and $\delta \geq 0$, let $Q$, $\hat{Q}$, and $m \geq 1$ be such that

$$\|\hat{Q} - F^m_{J,\nu} Q\|_\infty \leq \delta,$$

and let $\hat{J}$ be defined by

$$\hat{J}(i) = \min_{u \in U(i)} \hat{Q}(i,u), \qquad \forall\ i.$$

Then,

$$\|\hat{J} - J^*\|_\infty \le \|\hat{Q} - Q^*\|_\infty \le \alpha \max\big\{\|J - J^*\|_\infty, \|Q - Q^*\|_\infty\big\} + \delta.$$

**Proof:** Using the triangle inequality, the fact $Q^* = F^m_{J^*,\nu} Q^*$, and Lemma 5.1, we have

$$\|\hat{Q} - Q^*\|_\infty - \|\hat{Q} - F^m_{J,\nu} Q\|_\infty \le \|F^m_{J,\nu} Q - Q^*\|_\infty \le \alpha \max\big\{\|J - J^*\|_\infty, \|Q - Q^*\|_\infty\big\},$$

which together with the assumption $\|\hat{Q} - F^m_{J,\nu} Q\|_\infty \le \delta$, implies the right-hand side of the desired inequality. The left-hand side follows from the generic inequality (2.13). **Q.E.D.**

For any policy $\mu$, we denote by $T_\mu$ the mapping defined by

$$(T_\mu J)(i) = \sum_{j=1}^n p_{ij}\big(\mu(i)\big)\Big(g\big(i, \mu(i), j\big) + \alpha J(j)\Big), \qquad \forall\ i.$$

**Lemma 5.3:** Given $Q$, let $\mu$ be a policy such that $\mu(i)$ attains the minimum of $Q(i, u)$ over $u \in U(i)$, for all $i$. Then,

$$\|T_\mu J^* - J^*\|_\infty \le 2\|Q - Q^*\|_\infty.$$

**Proof:** Let $\beta = \|Q - Q^*\|_\infty$, and let $\mu^*$ be an optimal policy. We have for all $i$,

$$\big|Q\big(i, \mu(i)\big) - Q^*\big(i, \mu(i)\big)\big| \le \beta, \qquad \big|Q\big(i, \mu^*(i)\big) - Q^*\big(i, \mu^*(i)\big)\big| \le \beta.$$

Note that

$$Q^*\big(i, \mu(i)\big) = (T_\mu J^*)(i), \qquad Q^*\big(i, \mu^*(i)\big) = (T_{\mu^*} J^*)(i) = J^*(i),$$

and by the definition of $\mu$,

$$Q\big(i, \mu(i)\big) \le Q\big(i, \mu^*(i)\big).$$

Combining these relations, we have for all $i$,

$$(T_\mu J^*)(i) - J^*(i) \le (T_\mu J^*)(i) - Q\big(i, \mu(i)\big) + Q\big(i, \mu^*(i)\big) - J^*(i) \le \beta + \beta = 2\beta,$$

from which the desired inequality follows. **Q.E.D.**

**Proof of Prop. 5.1:** Let

$$\beta_k = \max\big\{\|J_k - J^*\|_\infty, \|Q_k - Q^*\|_\infty\big\}.$$

By applying Lemma 5.2 with $J = J_k$, $\hat{J} = J_{k+1}$, $Q = Q_k$, $\hat{Q} = Q_{k+1}$, we have

$$\|J_{k+1} - J^*\|_\infty \le \|Q_{k+1} - Q^*\|_\infty \le \alpha \max\big\{\|J_k - J^*\|_\infty, \|Q_k - Q^*\|_\infty\big\} + \delta = \alpha\beta_k + \delta. \tag{5.4}$$

By taking the maximum of the two leftmost terms, this relation also implies that

$$\beta_{k+1} \le \alpha\beta_k + \delta,$$

and by iteration

$$\beta_{k+1} \le \alpha^{k+1}\beta_0 + (\alpha^k + \alpha^{k-1} + \cdots + 1)\delta. \tag{5.5}$$

From Eq. (5.4) and Lemma 5.3,

$$\|T_{\mu_{k+1}} J^* - J^*\|_\infty \le 2\big(\alpha\beta_k + \delta\big).$$

Taking limit along a subsequence of $\mu_k$ that converges to a stationary policy $\mu$, we obtain

$$\|T_\mu J^* - J^*\|_\infty \le 2 \limsup_{k\to\infty} \big(\alpha\beta_k + \delta\big) \le \frac{2\delta}{1-\alpha},$$

where the second inequality follows from Eq. (5.5). We also have

$$\begin{aligned}\|T_\mu^k J^* - J^*\|_\infty &\le \|T_\mu^k J^* - T_\mu^{k-1} J^*\|_\infty + \|T_\mu^{k-1} J^* - T_\mu^{k-2} J^*\|_\infty + \cdots + \|T_\mu J^* - J^*\|_\infty \\ &\le (\alpha^{k-1} + \alpha^{k-2} + \cdots + 1)\|T_\mu J^* - J^*\|_\infty.\end{aligned}$$

Combining the last two relations, taking limit as $k \to \infty$, and using the fact $T_\mu^k J^* \to J_\mu$, we obtain

$$\|J_\mu - J^*\|_\infty \le \frac{2\delta}{(1-\alpha)^2}.$$

**Q.E.D.**

## 6. APPROXIMATION ALGORITHMS

In this section, we provide some details on how to combine the approximation scheme of Section 5 with Q-factor approximations and simulation-based methods that use low-dimensional calculations. In particular, we discuss algorithms for constructing approximations $Q_{k+1}$ to $Q_{J_k,\nu_k}$ or to $F_{J_k,\nu_k}^{m_k} Q_k$ [cf. Eq. (5.1)], which can be combined with the updating rule of Eq. (5.2),

$$J_{k+1}(i) = \min_{u\in U(i)} Q_{k+1}(i,u), \qquad \forall\ i, \tag{6.1}$$

and with some method to select $\nu_{k+1}$. These algorithms can also be viewed as approximation counterparts of specific cases of the lookup-table-based stochastic policy iteration Algorithm I, given in Section 4. The error bound of Prop. 5.1 holds for such schemes (although the constant $\delta$ is generally unknown). We first focus on the algorithm of Tsitsiklis and Van Roy [TsV99], which can be used for solving approximately optimal stopping problems. This algorithm obtains a Q-factor vector $\hat{Q}$ that approximates a fixed point $Q_{J,\nu}$, in place of the "policy evaluation" step (2.8) of the algorithm, and belongs to the class of projected equation methods (see e.g., [Ber07], [BeY09]).

For a given $J$ and $\nu$, we view $Q_{J,\nu}(i,u)$ as the Q-factor of the optimal stopping problem described in Section 2, which corresponds to the action of not stopping at pair $(i,u)$. We approximate $Q_{J,\nu}(i,u)$ using a linear approximation architecture of the form

$$\hat{Q}(i,u) = \phi(i,u)'r, \qquad \forall\ (i,u). \tag{6.2}$$

Here, $\phi(i,u)'$ is a row vector of $s$ features whose inner product $\hat{Q}(i,u)$ with a column vector of weights $r \in \Re^s$ provides a Q-factor approximation for $(i,u)$. We may view $\phi(i,u)$ as forming an $n \times s$ matrix whose columns are basis functions for a subspace within which Q-factor vectors are approximated. We do not discuss the important issue of selection of $\phi(i,u)$, but we note the possibility of its optimal choice within some restricted class by using gradient and random search algorithms (see Menache, Mannor, and Shimkin [MMS06], and Yu and Bertsekas [YuB09] for recent work on this subject).

For the typical policy evaluation cycle, we have an estimate of optimal cost

$$J(i) = \min_{u \in U(i)} \phi(i,u)'r_0, \qquad \forall\ i,$$

where $r_0$ is the weight vector obtained at the end of the preceding policy evaluation cycle ($J$ may be arbitrarily chosen for the first cycle). We select a randomized policy $\nu$, and we generate a single infinitely long simulated trajectory $\big\{(i_0,u_0),(i_1,u_1),\dots\big\}$ corresponding to an unstopped system, i.e., using transition probabilities from $(i_t,u_t)$ to $(i_{t+1},u_{t+1})$ given by

$$p_{i_t i_{t+1}}(u_t)\nu(u_{t+1} \mid i_{t+1}).$$

Following the transition $\big((i_t,u_t),(i_{t+1},u_{t+1})\big)$, we update $r_t$ by

$$r_{t+1} = r_t - \gamma_t \phi(i_t,u_t) q_t, \tag{6.3}$$

where $q_t$ is the temporal difference

$$q_t = \phi(i_t,u_t)'r_t - g(i_t,u_t,i_{t+1}) - \alpha \min\big\{J(i_{t+1}), \phi(i_{t+1},u_{t+1})'r_t\big\}, \tag{6.4}$$

and $\gamma_t$ is a positive stepsize that diminishes to 0.

For convergence the stepsize $\gamma_t$ must satisfy some conditions that are standard for stochastic approximation-type algorithms [e.g., $\gamma_t = O(1/t)$; see [TsV99]]. Assuming that these and some other technical conditions are satisfied [such as a full-rank assumption for the matrix formed by $\phi(i,u)$], Tsitsiklis and Van Roy [TsV99] show the convergence of $\{r_t\}$ to a vector $r^*$ such that $\phi(i,u)'r^*$ is the solution of a projected equation that is characteristic of the TD methodology. They also provide a bound on the error $\phi(i,u)'r^* - Q_{J,\nu}(i,u)$; see also Van Roy [Van09].

The preceding algorithm describes how to obtain an approximation $\hat{Q}_k$ to $Q_{J_k,\nu_k}$. Combined with the update rule (6.1), it yields an approximate policy iteration method, where exploration is encoded in the choice of $\nu_k$ (which can be selected arbitrarily). The convergence properties of this method may be quite complicated, not only because $\hat{Q}_k$ is just an approximation to $Q_{J_k,\nu_k}$, but also because when Q-factor approximations of the form (6.2) are used, policy oscillations may occur, a phenomenon described in Section 6.4 of [BeT96] (see also [Ber10], Section 6.3).

We note a related scaled version of the algorithm (6.3), proposed by Choi and Van Roy [ChV06]:

$$r_{t+1} = r_t - \gamma_t D_t^{-1}\phi(i_t,u_t)q_t, \tag{6.5}$$

where $D_t$ is a positive definite scaling matrix. For our purposes, to keep overhead per iteration low, it is important that $D_t$ is chosen to be diagonal, and [ChV96] suggests suitable simulation-based choices. We also note alternative iterative optimal stopping algorithms given by Yu and Bertsekas [YuB07], which have faster convergence properties, but require more overhead per iteration because they require a sum of past temporal differences in the right-hand side of Eq. (6.5).

The preceding algorithms require an infinitely long trajectory $\big\{(i_0,u_0),(i_1,u_1),\ldots\big\}$ for convergence. In the context of our policy iteration algorithm, however, it may be important to use finitely long and even short trajectories between updates of $J_k$ and $\nu_k$. This is consistent with the ideas of optimistic policy iteration (explained for example in [BeT96], [SuB98], [Ber07], [Ber10]; for recent experimental studies, see Jung and Polani [JuP07], and Busoniu et al. [BED09]). It is also suggested by the value iteration nature of the lookup table version of the algorithm when $\nu_k$ involves a substantial amount of exploration, as explained in Section 2. Some experimentation with optimistic methods should be helpful in clarifying the associated issues.

## 7. CONCLUSIONS

We have developed a policy iteration algorithm for Q-learning in discounted MDP. In its lookup table form, the algorithm admits interesting asynchronous and optimistic implementations, with sound convergence properties. In its compact representation/approximate form, the algorithm addresses in a new way the critical issue of exploration in the context of simulation-based approximations using TD methods.

## 8. REFERENCES

[ABB02] Abounadi, J., Bertsekas, D. P., and Borkar, V., "Stochastic Approximation for Non-Expansive Maps: Application to Q-Learning Algorithms," SIAM J. on Control and Optimization, Vol. 41, pp. 1-22.

[BED09] Busoniu, L., Ernst, D., De Schutter, B., and Babuska, R., 2009. "Online Least-Squares Policy Iteration for Reinforcement Learning Control," unpublished report, Delft Univ. of Technology, Delft, NL.

[BeI96] Bertsekas, D. P., and Ioffe, S., 1996. "Temporal Differences-Based Policy Iteration and Applications in Neuro-Dynamic Programming," Lab. for Info. and Decision Systems Report LIDS-P-2349, MIT, Cambridge, MA.

[BeT89] Bertsekas, D. P., and Tsitsiklis, J. N., 1989. Parallel and Distributed Computation: Numerical Methods, Prentice-Hall, Englewood Cliffs, N. J; republished by Athena Scientific, Belmont, MA, 1997.

[BeT96] Bertsekas, D. P., and Tsitsiklis, J. N., 1996. Neuro-Dynamic Programming, Athena Scientific, Belmont, MA.

[Ber82] Bertsekas, D. P., 1982. "Distributed Dynamic Programming," IEEE Trans. Automatic Control, Vol. AC-27, pp. 610-616.

[Ber83] Bertsekas, D. P., 1983. "Asynchronous Distributed Computation of Fixed Points," Math. Programming, Vol. 27, pp. 107-120.

[Ber05] Bertsekas, D. P., 2005. Dynamic Programming and Optimal Control, 3rd Edition, Vol. I, Athena Scientific, Belmont, MA.

[Ber07] Bertsekas, D. P., 2007. Dynamic Programming and Optimal Control, 3rd Edition, Vol. II, Athena Scientific, Belmont, MA.

[Ber10] Bertsekas, D. P., 2010. Approximate Dynamic Programming, on-line at
http://web.mit.edu/dimitrib/www/dpchapter.html.

[Bor98] Borkar, V. S., 1998. "Asynchronous Stochastic Approximations," SIAM J. on Control and Optimization, Vol. 36, pp. 840-851; correction note in *ibid.*, Vol. 38, pp. 662-663.

[Bor08] Borkar, V. S., 2008. Stochastic Approximation: A Dynamical Systems Viewpoint, Cambridge Univ. Press, N. Y.

[Boy02] Boyan, J. A., 2002. "Technical Update: Least-Squares Temporal Difference Learning," Machine Learning, Vol. 49, pp. 1-15.

[BrB96] Bradtke, S. J., and Barto, A. G., 1996. "Linear Least-Squares Algorithms for Temporal Difference Learning," Machine Learning, Vol. 22, pp. 33-57.

[CFH07] Chang, H. S., Fu, M. C., Hu, J., Marcus, S. I., 2007. Simulation-Based Algorithms for Markov Decision Processes, Springer, N. Y.

[Cao07] Cao, X. R., 2007. Stochastic Learning and Optimization: A Sensitivity-Based Approach, Springer, N. Y.

[ChV06] Choi, D. S., and Van Roy, B., 2006. "A Generalized Kalman Filter for Fixed Point Approximation and Efficient Temporal-Difference Learning," Discrete Event Dynamic Systems: Theory and Applications, Vol. 16, pp. 207-239.

[Gor95] Gordon, G. J., 1995. "Stable Function Approximation in Dynamic Programming," in Machine Learning: Proceedings of the Twelfth International Conference, Morgan Kaufmann, San Francisco, CA.

[Gos03] Gosavi, A., 2003. Simulation-Based Optimization Parametric Optimization Techniques and Reinforcement Learning, Springer-Verlag, N. Y.

[JJS94] Jaakkola, T., Jordan, M. I., and Singh, S. P., 1994. "On the Convergence of Stochastic Iterative Dynamic Programming Algorithms," Neural Computation, Vol. 6, pp. 1185-1201.

[JSJ95] Jaakkola, T., Singh, S. P., and Jordan, M. I., 1995. "Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems," Advances in Neural Information Processing Systems, Vol. 7, pp. 345-352.

[JuP07] Jung, T., and Polani, D., 2007. "Kernelizing LSPE($\lambda$)," in Proc. 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning, Honolulu, Hawaii. pp. 338-345.

[MMS06] Menache, I., Mannor, S., and Shimkin, N., 2005. "Basis Function Adaptation in Temporal Difference Reinforcement Learning," Ann. Oper. Res., Vol. 134, pp. 215-238.

[MSB08] Maei, H. R., Szepesvari, C., Bhatnagar, S., Silver, D., Precup, D., and Sutton, R. S., 2009. "Convergent Temporal-Difference Learning with Arbitrary Smooth Function Approximation," Proc. NIPS.

[Mey07] Meyn, S., 2007. Control Techniques for Complex Networks, Cambridge University Press, N. Y.

[Pow07] Powell, W. B., 2007. Approximate Dynamic Programming: Solving the Curses of Dimensionality, Wiley, N. Y.

[Put94] Puterman, M. L., 1994. Markov Decision Processes: Discrete Stochastic Dynamic Programming, J. Wiley, N. Y.

[SMP09] Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvari, C., and Wiewiora, E., 2009. "Fast Gradient-Descent Methods for Temporal-Difference Learning with Linear Function Approximation," Proc. of ICML.

[SSM08] Sutton, R. S., Szepesvari, C., and Maei, H. R., 2008. "A Convergent O(n) Algorithm for Off-Policy Temporal-Difference Learning with Linear Function Approximation," Proc. of NIPS 21.

[SuB98] Sutton, R. S., and Barto, A. G., 1998. Reinforcement Learning, MIT Press, Cambridge, MA.

[Sut88] Sutton, R. S., 1988. "Learning to Predict by the Methods of Temporal Differences," Machine Learning, Vol. 3, pp. 9-44.

[TBA86] Tsitsiklis, J. N., Bertsekas, D. P., and Athans, M., 1986. "Distributed Asynchronous Deterministic and Stochastic Gradient Optimization Algorithms," IEEE Trans. on Aut. Control, Vol. AC-31, pp. 803-812.

[TsV96] Tsitsiklis, J. N., and Van Roy, B., 1996. “Feature-Based Methods for Large-Scale Dynamic Programming,” Machine Learning, Vol. 22, pp. 59-94.

[TsV99] Tsitsiklis, J. N., and Van Roy, B., 1999. “Optimal Stopping of Markov Processes: Hilbert Space Theory, Approximation Algorithms, and an Application to Pricing Financial Derivatives,” IEEE Transactions on Automatic Control, Vol. 44, pp. 1840-1851.

[Tsi94] Tsitsiklis, J. N., 1994. “Asynchronous Stochastic Approximation and Q-Learning,” Machine Learning, Vol. 16, pp. 185-202.

[Tsi02] Tsitsiklis, J. N., 2002. “On the Convergence of Optimistic Policy Iteration,” J. of Machine Learning Research, Vol. 3, pp. 59-72.

[Van09] Van Roy, B., 2009. “On Regression-Based Stopping Times,” Discrete Event Dynamic Systems, to appear.

[Wat89] Watkins, C. J. C. H., Learning from Delayed Rewards, Ph.D. Thesis, Cambridge Univ., England.

[WiB93] Williams, R. J., and Baird, L. C., 1993. “Analysis of Some Incremental Variants of Policy Iteration: First Steps Toward Understanding Actor-Critic Learning Systems,” Report NU-CCS-93-11, College of Computer Science, Northeastern University, Boston, MA.

[YuB07] Yu, H., and Bertsekas, D. P., 2007. “A Least Squares Q-Learning Algorithm for Optimal Stopping Problems,” Lab. for Information and Decision Systems Report 2731, MIT; also in Proc. European Control Conference 2007, Kos, Greece.

[YuB09] Yu, H., and Bertsekas, D. P., 2009. “Basis Function Adaptation Methods for Cost Approximation in MDP,” Proc. of 2009 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning, Nashville, Tenn.