

Q-Learning and Enhanced Policy Iteration in Discounted Dynamic Programming

Dimitri P. Bertsekas and Huizhen Yu

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 involves (possibly inexact) solution of an optimal stopping problem. This problem 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 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.

I. INTRODUCTION

We consider the approximate solution of large-scale discounted infinite horizon dynamic programming (DP) problems. The states are denoted $i = 1, \dots, n$. 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 stationary policies μ 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 μ over an infinite number of stages starting from state i , and by $J^*(i)$ the minimal value of $J_\mu(i)$ over all μ . We denote by J_μ and J^* the vectors of \mathbb{R}^n (n -dimensional space) with components $J_\mu(i)$ and $J^*(i)$, $i = 1, \dots, n$, respectively. This is the standard discounted Markovian decision problem (MDP) context, discussed in many sources (e.g., 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(λ) (Sutton [Sut88]), LSPE(λ) (Bertsekas and Ioffe [BeI96]), and LSTD(λ) (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 $\{(i_t, \mu(i_t)), t = 0, 1, \dots\}$ using the Markov chain corresponding to the current policy μ , which means that state-control pairs $(i, u) \neq (i, \mu(i))$ 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 μ . 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(λ) and TD(λ) 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(λ) 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(λ) 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

Dimitri Bertsekas was supported by NSF Grant ECCS-0801549. Huizhen Yu was supported in part by Academy of Finland Grant 118653 (ALGO-DAN) and the PASCAL Network of Excellence, IST-2002-506778.

Dimitri Bertsekas is with the Dept. of Electr. Engineering and Comp. Science, M.I.T., Cambridge, Mass., 02139. dimitrib@mit.edu

Huizhen Yu is with the Dept. of Computer Science, Univ. of Helsinki, Finland. janey.yu@cs.helsinki.fi

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.

This paper is organized as follows. In Section II, we introduce our policy iteration-like algorithm for the case of exact/lookup table representation of Q-factors, and ad-

dress convergence issues. In Section III, we show that our algorithm admits interesting asynchronous and optimistic versions that resemble both Q-learning and modified policy iteration, and have solid convergence properties. In Section IV, we consider the possibility of approximating the policy evaluation portion of our algorithm, and we provide a corresponding error bound. In this section, we also 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(λ) method for policy evaluation.

II. 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 $T : \mathbb{R}^n \mapsto \mathbb{R}^n$ given by

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

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

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

The optimal Q-factor vector Q^* is the unique fixed point of the mapping F defined for all (i, u) by

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

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 current policy μ , we find Q_μ , the unique fixed point of the mapping F_μ corresponding to μ , and given by

$$(F_\mu Q)(i, u) = \sum_{j=1}^n p_{ij}(u) \left(g(i, u, j) + \alpha Q(j, \mu(j)) \right), \quad \forall (i, u), \quad (2.3)$$

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

$$\bar{\mu}(i) = \arg \min_{u \in U(i)} Q_\mu(i, u), \quad \forall i, \quad (2.4)$$

(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_μ of (2.3) with another mapping that allows exploration as well as a dependence on μ . This mapping, denoted $F_{J,\nu}$, depends on a vector $J \in \mathbb{R}^n$, with components denoted $J(i)$, and on a randomized policy ν , which for each state i defines a probability distribution

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

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 for all (i, u) by

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

Comparing $F_{J,\nu}$ with the classical Q-learning mapping of (2.2) [or the mapping F_μ of (2.3)], we see that they take into account the Q-factors of the next state j differently: F (or F_μ) uses the minimal Q-factor $\min_{v \in U(j)} Q(j, v)$ [the Q-factor $Q(j, \mu(j))$, respectively], while $F_{J,\nu}$ uses a randomized Q-factor [according to $\nu(v | j)$], 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. (2.2)].

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

- The state space is the set of state-control pairs (i, u) of the original problem.
- When at state (i, u) , if we decide to stop, we incur a stopping cost $J(i)$ (independent of u).
- 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 $p_{ij}(u)\nu(v | j)$.

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 α for all ν and J , i.e.,

$$\|F_{J,\nu}Q - F_{J,\nu}\tilde{Q}\|_\infty \leq \alpha \|Q - \tilde{Q}\|_\infty, \quad \forall Q, \tilde{Q}, \quad (2.6)$$

where $\|\cdot\|_\infty$ denotes the sup-norm ($\|Q\|_\infty = \max_{(i,u)} |Q(i, u)|$). 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 π , 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 ν to following the policy π .

For a given (J, ν) , 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 at each iteration and does not require matrix inversion (like LSTD or LSPE). Later, in Sections IV, we will envision the use of this algorithm for approximating $Q_{J,\nu}$.

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

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

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

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

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

Proof: We write

$$F_{J,\nu}Q = \bar{g} + \alpha \bar{P}_\nu \min\{J^x, Q\}, \quad (2.7)$$

where \bar{g} is the vector with components

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

and \bar{P}_ν is the transition probability matrix with probabilities of transition $(i, u) \rightarrow (j, v)$ equal to

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

From (2.7), we obtain

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

We also have¹

$$\begin{aligned} & \|\min\{J^x, Q\} - \min\{\tilde{J}^x, \tilde{Q}\}\|_\infty \\ & \leq \max\{\|J - \tilde{J}\|_\infty, \|Q - \tilde{Q}\|_\infty\}. \end{aligned}$$

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 ν_k and an arbitrary positive integer m_k , and we obtain the next pair (Q_{k+1}, J_{k+1}) as follows:

¹Here we are using a nonexpansiveness property of the minimization map: for any $Q_1, Q_2, \tilde{Q}_1, \tilde{Q}_2$, we have

$$\begin{aligned} & \|\min\{Q_1, Q_2\} - \min\{\tilde{Q}_1, \tilde{Q}_2\}\|_\infty \\ & \leq \max\{\|Q_1 - \tilde{Q}_1\|_\infty, \|Q_2 - \tilde{Q}_2\|_\infty\}. \end{aligned}$$

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

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

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

Iteration k with Lookup Table Representation:

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

$$Q_{k+1} = F_{J_k, \nu_k}^{m_k} Q_k. \quad (2.8)$$

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

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

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

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

so (2.8) coincides with the synchronous Q-learning algorithm $Q_{k+1} = FQ_k$, while (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, ν_k} of the associated stopping problem (the unique fixed point of F_{J_k, ν_k}), and the algorithm takes the form

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

Assume further that ν_k is chosen to be the deterministic policy μ_k that attains the minimum in the equation

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

with ν_0 being some deterministic policy μ_0 satisfying $J_0 \geq J_{\mu_0}$. Then Q_1 is equal to Q_{J_0, μ_0} (since $m_k = \infty$) and can be seen to be also equal to the (exact) Q-factor vector of μ_0 (since $J_0 \geq J_{\mu_0}$), so μ_1 as generated by (2.11), is the policy generated from μ_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 ν_k may differ significantly from the deterministic policy (2.11). In particular, suppose that $m_k = \infty$ and ν_k is chosen to assign positive probability to nonoptimal controls, i.e., so that $\nu_k(\mu^*(j) | j) = 0$ for all j and optimal policies μ^* . Then since $J_k \rightarrow J^*$ (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 $\nu_k(v | j) > 0$, so that

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

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)!

The preceding arguments illustrate that the choices of ν_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 ν_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 $\nu_k(\mu_k(j) | j) = 0$ for any policy μ_k generated according to (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. Section IV), a reasonable way to operate the algorithm is to determine ν_k by “superimposing” some exploration to the deterministic policy μ_k of (2.11). For example, we may use a distribution ν_k that is a random mixture of μ_k and another policy that induces exploration, including visits to state-control pairs that are unlikely/impossible to generate under μ_k . In this case, we may view the calculation of Q_{k+1} via (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 ν_k and the value of m_k , and we may view (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 $\{m_k\}$, we have*

$$\lim_{k \rightarrow \infty} Q_k = Q^*, \quad \lim_{k \rightarrow \infty} J_k = J^*,$$

and the rate of convergence is geometric. Furthermore, for all k after some index \bar{k} , the generated policies μ_k are optimal.

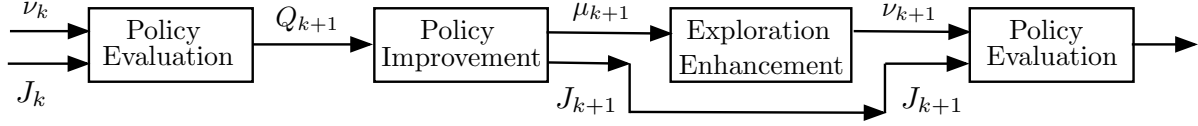


Fig. 2.1. Illustration of exploration-enhanced policy iteration algorithm. The policy evaluation consists of a finite number of Q-value iterations for the optimal stopping problem involving the randomized policy ν and the threshold/stopping cost J [cf. (2.8)]. It is followed by policy improvement that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

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

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

Using this relation, we have

$$\begin{aligned} \|F_{J_k, \nu_k}^2 Q_k - Q^*\|_\infty &\leq \alpha \max \{ \|J_k - J^*\|_\infty, \|F_{J_k, \nu_k} Q_k - Q^*\|_\infty \} \\ &\leq \max \{ \alpha \|J_k - J^*\|_\infty, \alpha^2 \|Q_k - Q^*\|_\infty \}, \end{aligned}$$

and by repeating this process,

$$\begin{aligned} \|Q_{k+1} - Q^*\|_\infty &= \|F_{J_k, \nu_k}^{m_k} Q_k - Q^*\|_\infty \\ &\leq \max \{ \alpha \|J_k - J^*\|_\infty, \alpha^{m_k} \|Q_k - Q^*\|_\infty \}. \end{aligned} \quad (2.12)$$

Since for all Q , and \tilde{Q} , we have²

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

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

$$\|J_k - J^*\|_\infty \leq \|Q_k - Q^*\|_\infty. \quad (2.14)$$

Combining (2.12) and (2.14), we obtain

$$\|Q_{k+1} - Q^*\|_\infty \leq \alpha \|Q_k - Q^*\|_\infty. \quad (2.15)$$

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

III. ASYNCHRONOUS AND STOCHASTIC VERSIONS

The algorithm, as given in (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 (2.13)] can be used to establish the convergence of the algorithm under far more irregular conditions. In particular, we may consider asynchronous updating of Q-factors and state costs corresponding to blocks of components, and

²This is a well-known property. For a proof, write

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

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

also model-free sampled versions, which do not require the explicit knowledge of $p_{ij}(u)$ and the calculation of expected values.

In an asynchronous version of the standard policy iteration for Q-factors [cf. (2.4)], the updates of μ and Q are executed selectively, for only some of the states and state-control pairs. In such an algorithm, 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_μ where μ is updated. In a fairly general implementation discussed in the literature ([BeT96], Section 2.2, or [Ber07], Section 1.3.3), the algorithm generates a sequence of pairs (Q_k, μ_k) , starting from an arbitrary pair (Q_0, μ_0) as follows:

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

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

where one of the two sets R_k or S_k is nonempty and the other set is empty. Relative to ordinary Q-learning, the advantage is that the minimization in (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 μ_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 (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 μ that is maintained and updated selectively at some of the states. This policy may be used to generate a randomized policy ν which enters the algorithm in a

material way. However, the algorithm is valid for any choice of ν , so its definition need not involve the policy μ and the method in which it is used to update ν (we will later give an example of an updating scheme for μ and ν). 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 distribution ν_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

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

and generate J_{k+1} according to

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

As mentioned earlier, the preceding algorithm as stated does not have the form of policy iteration. However, ν_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 (3.1)-(3.2). For an example of such an algorithm, which has a character similar to the algorithm (3.1)-(3.2), assume that a policy μ_k is also maintained, which defines ν_k (so ν_k is the deterministic policy μ_k). Let there be two types of iterations: those corresponding to an index subset K_Q where Q is updated, and those corresponding to the complementary subset K_μ where μ and J are updated. We assume that K_Q and K_μ are infinite. It is not essential that K_Q and K_μ be determined in advance; they may instead depend on algorithmic progress according to some scheme. An arbitrary sequence of state-control pairs $\{(i_k, u_k) \mid k \in K_Q\}$ is generated subject to the condition that each state-control pair appears in this sequence infinitely often. For $k \in K_Q$, the algorithm updates just the (i_k, u_k) th component of Q according to

$$Q_{k+1}(i, u) = \begin{cases} (F_{J_k, \mu_k} Q_k)(i, u) & \text{if } (i, u) = (i_k, u_k), \\ Q_k(i, u) & \text{if } (i, u) \neq (i_k, u_k). \end{cases} \quad (3.5)$$

For $k \in K_\mu$, the algorithm updates J_k and μ_k for all j in the set

$$S_k = \{j \mid p_{i_{k+1}j}(u_{k+1}) > 0\},$$

according to

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

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

We may view (3.5) as a policy evaluation iteration for the single state-control pair (i_k, u_k) , and (3.6) as a policy

improvement iteration only for the states in S_k . In comparing our algorithm (3.5)-(3.6) with the known algorithm (3.1)-(3.2), we see that the essential difference is the use of a Q-learning/optimal stopping iteration (3.5) for policy evaluation, in place of the ordinary policy evaluation iteration for Q-factors (3.1). Assuming that each state is contained in the sets S_k infinitely often, we will see that the algorithm (3.5)-(3.6) fulfills the conditions of the following proposition, and is convergent to (Q^*, J^*) , in contrast with the asynchronous policy iteration (3.1)-(3.2), which requires an additional restriction, such as $F_{\mu_0} Q_0 \leq Q_0$, as noted earlier. The stronger convergence property of our algorithm may be attributed to the fact that it uses a somewhat different mechanism for convergence, which relies on an underlying sup-norm contraction (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). A proof is given in our extended report [BeY10].

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 $\{(Q_k, J_k)\}$ generated by the algorithm (3.3)-(3.4) converges to (Q^*, J^*) .*

We will now consider a stochastic iterative version of our algorithm, which is 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 $\{(i_k, u_k) \mid k = 0, 1, \dots\}$ by any probabilistic mechanism that guarantees that each pair (i, u) appears infinitely often with probability 1, we generate a successor state j_k according to the distribution $p_{i_k j}(u_k)$, $j = 1, \dots, n$, and we update only the Q-factor of (i_k, u_k) at iteration k ,

$$Q_{k+1}(i_k, u_k) = (1 - \gamma_k) Q_k(i_k, u_k) + \gamma_k \left(g(i_k, u_k, j_k) + \alpha \min_{v \in U(j)} Q_k(j_k, v) \right), \quad (3.7)$$

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 stepsize sequence $\{\gamma_k\}$ must diminish to 0 at a suitable rate [e.g., $O(1/k)$], and must satisfy assumptions that are standard in stochastic approximation methods. There are also asynchronous versions of the algorithm (3.7), where $Q_k(j_k, v)$ may be replaced by $Q_{k-\tau_{k,v}}(j_k, v)$, where $\tau_{k,v}$ 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].

We present a simple version of our algorithm; the extended report [BeY10] contains more general versions with a comprehensive analysis. This new algorithm similarly generates a sequence of state-control pairs $\{(i_k, u_k) \mid k = 0, 1, \dots\}$, and updates only the Q-factor of (i_k, u_k) at iteration k ,

using a positive stepsize $\gamma_k(i_k, u_k)$. 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 ν_k arbitrarily for each k , but it may maintain a policy μ_k that is updated at selected states simultaneously with J , and then use $\nu_k = \mu_k$; this type of choice of ν_k gives the algorithm a modified policy iteration character. Compared to the preceding Q-learning algorithm (3.7), 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:

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

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

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

- (2) Select a stepsize $\gamma_k(i_k, u_k) \in (0, 1]$ and a policy ν_k . Generate a successor state j_k according to the distribution $p_{i_k j}(u_k)$, $j = 1, \dots, n$, and generate a control v_k according to the distribution $\nu_k(v | j_k)$, $v \in U(j_k)$.
- (3) Update the (i_k, u_k) th component of Q according to

$$\begin{aligned} Q_{k+1}(i_k, u_k) = & (1 - \gamma_k(i_k, u_k))Q_k(i_k, u_k) \\ & + \gamma_k(i_k, u_k) \left(g(i_k, u_k, j_k) \right. \\ & \left. + \alpha \min\{J_{k+1}(j_k), Q_k(j_k, v_k)\} \right), \end{aligned} \quad (3.8)$$

and leave all other components of Q_k unchanged:

$$Q_{k+1}(i, u) = Q_k(i, u) \text{ for all } (i, u) \neq (i_k, u_k).$$

The convergence of the preceding algorithm can be established by using the asynchronous stochastic approximation-type arguments of Tsitsiklis [Tsi94]. We refer to the extended report [BeY10] for the proof.

Proposition 3.2: *Assume that with probability 1 the following hold: each pair (i, u) is selected infinitely often; for each state i , the set $\{k | i_k = i, k \in K_J\}$ is infinite; and for each pair (i, u) ,*

$$\sum_{k: (i_k, u_k) = (i, u)} \gamma_k(i, u) = \infty, \quad \sum_{k: (i_k, u_k) = (i, u)} \gamma_k^2(i, u) \leq C$$

for some deterministic constant C . Then any sequence $\{(Q_k, J_k)\}$ generated by the model-free optimistic policy iteration algorithm converges to (Q^*, J^*) with probability 1.

A simple way to choose stepsize sequences $\{\gamma_k(i, u)\}$ that satisfy the condition in the proposition is to define them using a positive scalar sequence $\{\gamma_k\}$ which diminishes to 0 at a suitable rate [e.g., $O(1/k)$]. Let $\gamma_k(i, u) = \gamma_k$ if $(i, u) =$

(i_k, u_k) and let $\gamma_k(i, u) = 0$ otherwise, and select (i, u) “comparably often” in the sense that the fraction of times (i, u) is selected is nonzero in the limit (see Borkar [Bor08]).

Note that our algorithm also has similarities with the partially optimistic TD(0) algorithm, discussed in Section 5.4 of [BeT96]. The latter algorithm updates J (rather than 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]).

IV. APPROXIMATIONS AND ERROR BOUNDS

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

$$\|Q_{k+1} - F_{J_k, \nu_k}^{m_k} Q_k\|_\infty \leq \delta, \quad (4.1)$$

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

$$J_{k+1}(i) = \min_{u \in U(i)} Q_{k+1}(i, u), \quad \forall i, \quad (4.2)$$

and let the randomized policy ν_{k+1} be arbitrary as before. The following error bound is identical to what is generally viewed as the standard bound for the performance of approximate policy iteration ([BeT96], Prop. 6.2). The analysis also holds when m_k may be equal to ∞ ; a proof is given in our extended report [BeY10].

Proposition 4.1: *Assume that for some $\delta \geq 0$ and each $k \geq 0$, there exists a positive integer m_k such that (4.1) holds. Let μ_{k+1} be a policy such that $\mu_{k+1}(i)$ attains the minimum in (4.2) for all i . Then, for any stationary policy μ that is a limit point of $\{\mu_k\}$, we have*

$$\|J_\mu - J^*\|_\infty \leq \frac{2\delta}{(1 - \alpha)^2}. \quad (4.3)$$

The computation of Q_{k+1} can be done in a number of ways. We discuss here simulation-based approximation methods that use low-dimensional calculations. For a given J and ν , we view $Q_{J, \nu}(i, u)$ as the Q-factor of the optimal stopping problem described in Section II, 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

$$\hat{Q}(i, u) = \phi(i, u)'r, \quad \forall (i, u). \quad (4.4)$$

Here, $\phi(i, u)'$ is a row vector of s features whose inner product $\hat{Q}(i, u)$ with a column vector of weights $r \in \mathbb{R}^s$ 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. Using a single infinitely long simulated trajectory $\{(i_0, u_0), (i_1, u_1), \dots\}$ corresponding to an un-stopped system, the algorithm of Tsitsiklis and Van Roy

[TsV99] calculates approximate Q-factors $\phi(i, u)'r^*$, which correspond to the solution of a projected equation that is characteristic of the TD methodology. This algorithm does not require matrix inversion, which may be an advantage for problems with a large number of features s . Variants of this algorithm include the ones of Choi and Van Roy [ChV06], Yu and Bertsekas [YuB07]. Tsitsiklis and Van Roy [TsV99] also provide a bound on the error $\phi(i, u)'r^* - Q_{J, \nu}(i, u)$; see also Van Roy [Van09].

As to the important issue of selection of $\phi(i, u)$, we note the possibility of its optimal choice within some restricted class by using gradient and random search algorithms (see Menache, Mannor, and Shimkin [MMS05], and Yu and Bertsekas [YuB09] for recent work on this subject).

V. 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.

REFERENCES

- [ABB02] Abounadi, J., Bertsekas, D. P., and Borkar, V., 2002. "Stochastic Approximation for Non-Expansive Maps: Application to Q-Learning Algorithms," *SIAM J. on Control and Optimization*, Vol. 41, pp. 1-22.
- [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.
- [BeY10] Bertsekas, D. P., and Yu, H., 2010. "Q-Learning and Enhanced Policy Iteration in Discounted Dynamic Programming" Lab. for Information and Decision Systems Report 2831, MIT.
- [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, online at <http://web.mit.edu/dimitrib/www/dpchapter.html>.
- [BhB08] Bhatnagar, S., and Babu, K. M., 2008. "New Algorithms of the Q-Learning Type," *Automatica*, Vol. 44, pp. 1111-1119.
- [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*, Hindustan Book Agency, New Delhi, India.
- [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*, Vol. 16, pp. 207-239.
- [Gor95] Gordon, G. J., 1995. "Stable Function Approximation in Dynamic Programming," *Proc. ICML*.
- [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," *Proc. NIPS*.
- [MMS05] 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*, 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. 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. NIPS*.
- [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 Trans. on Aut. 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. IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL 2009)*, Nashville, Tenn.