August 1996 (Revised August 1997) **Report LIDS-P-2349**

Temporal Differences-Based Policy Iteration and Applications in Neuro-Dynamic Programming[1]

by

Dimitri P. Bertsekas[2] and Sergey Ioffe [3]

Abstract

We introduce a new policy iteration method for dynamic programming problems with discounted and undiscounted cost. The method is based on the notion of temporal differences, and is primarily geared to the case of large and complex problems where the use of approximations is essential. We develop the theory of the method without approximation, we describe how to embed it within a neuro-dynamic programming/reinforcement learning context where feature-based approximation architectures are used, we relate it to TD(λ) methods, and we illustrate its use in the training of a tetris playing program.

[1] Supported by the National Science Foundation under Grant DDM-8903385 and Grant CCR-9103804. Thanks are due to John Tsitsiklis for several helpful discussions and to Dimitris Papaioannou, who assisted with some of the experiments.

[2] Department of Electrical Engineering and Computer Science, M. I. T., Cambridge, Mass., 02139.

[3] Department of Electrical Engineering and Computer Science, M. I. T., Cambridge, Mass., 02139.

1. INTRODUCTION

We consider a standard discounted infinite horizon model in dynamic programming (DP for short). We are given a discrete-time dynamic system whose state transition depends on a control. We assume that there are n states, denoted by $1, 2, \ldots, n$. When at state i, the control must be chosen from a given finite set $U(i)$. At state i, the choice of a control u specifies the transition probability $p_{ij}(u)$ to the next state j. At the kth transition, we incur a cost $\alpha^k g(i, u, j)$, where g is a given function, and α is a discount factor with $0 < \alpha < 1$. We will later discuss extensions of our analysis to undiscounted problems.

We are interested in *policies*, that is, sequences $\pi = \{\mu_0, \mu_1, \ldots\}$ where each μ_k is a function mapping states into controls with $\mu_k(i) \in U(i)$ for all states i. Let us denote by i_k the state at time k. The total expected cost starting from an initial state i and using a policy $\pi = \{\mu_0, \mu_1, \ldots\}$ is

$$J^\pi(i) = \lim_{N\to\infty} E\left[\sum_{k=0}^{N-1} \alpha^k g\big(i_k, \mu_k(i_k), i_{k+1}\big) \;\Big|\; i_0 = i\right].$$

The optimal cost-to-go starting from state i is denoted by $J^*(i)$; that is,

$$J^*(i) = \min_\pi J^\pi(i).$$

We view the costs $J^*(i)$, $i = 1, \ldots, n$, as the components of a vector J^*, referred to as the *optimal cost-to-go vector*.

A stationary policy is a policy of the form $\pi = \{\mu, \mu, \ldots\}$. The corresponding cost-to-go is denoted by $J^\mu(i)$. For brevity, we refer to $\{\mu, \mu, \ldots\}$ as the stationary policy μ. We say that μ is optimal if $J^\mu(i) = J^*(i)$ for all states i. The vector J^μ that has components $J^\mu(i)$, $i = 1, \ldots, n$, is referred to as the *cost-to-go vector of the stationary policy* μ.

We introduce some notation. For any vector $J = \big(J(1), \ldots, J(n)\big)$, we consider the vector TJ obtained by applying one iteration of the DP algorithm to J; the components of TJ are

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

Similarly, for any vector J and any stationary policy μ, we consider the vector $T_\mu J$ with components

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

It is well-known that the mappings T and T_μ are contraction mappings with respect to the maximum norm. For easy reference, we summarize the main results for discounted problems in the following proposition (see e.g., [Ber95], [Put94], [Ros83]).

Proposition 1: The following hold for the discounted cost problem:

(a) The optimal cost-to-go vector J^* satisfies

$$J^* = TJ^*. \tag{1.1}$$

Furthermore, J^* is the only solution of this equation.

(b) We have

$$\lim_{t\to\infty} T^t J = J^*,$$

for every vector J.

(c) A stationary policy μ is optimal if and only if

$$T_\mu J^* = TJ^*.$$

(d) For a stationary policy μ, the associated cost-to-go vector J^μ satisfies

$$\lim_{t\to\infty} T_\mu^t J = J^\mu,$$

for every vector J. Furthermore,

$$J^\mu = T_\mu J^\mu,$$

and J^μ is the only solution of this equation.

In the policy iteration algorithm, we start with a stationary policy μ_0, and we generate a sequence of new stationary policies $\mu_1, \mu_2, \ldots$. Given the policy μ_t, we perform a *policy evaluation step*, that computes $J^{\mu_t}(i)$, $i = 1, \ldots, n$, as the solution of the (linear) system of equations $J = T_{\mu_t} J$ [cf. Prop. 1(d)], or equivalently

$$J(i) = \sum_{j=1}^{n} p_{ij}\big(\mu_t(i)\big)\Big(g\big(i, \mu_t(i), j\big) + \alpha J(j)\Big), \qquad i = 1, \ldots, n, \tag{1.2}$$

in the n unknowns $J(1), \ldots, J(n)$ [cf. Prop. 1(d)]. We then perform a *policy improvement step*, which computes a new policy μ_{t+1} as

$$\mu_{t+1}(i) = \arg\min_{u\in U(i)} \sum_{j=0}^{n} p_{ij}(u)\big(g(i,u,j) + J^{\mu_t}(j)\big), \qquad i = 1, \ldots, n, \tag{1.3}$$

or equivalently, as the policy satisfying

$$T_{\mu_{t+1}} J^{\mu_t} = TJ^{\mu_t}.$$

The process is repeated with μ_{t+1} used in place of μ_t, unless we have $J^{\mu_{t+1}}(i) = J^{\mu_t}(i)$ for all i, in which case the algorithm terminates with the policy μ_t. It is well-known that the policy iteration algorithm generates an improving sequence of policies [that is, $J^{\mu_{t+1}}(i) \leq J^{\mu_t}(i)$ for all i and t] and terminates with an optimal policy.

In this paper, we consider a new policy iteration method that is based on the notion of temporal differences, and is primarily motivated by the case of large and complex problems where the use of approximations is essential. We develop the theory of this method without any approximation, and we describe how to embed it within a simulation-based context where feature-based approximation architectures are used. We also relate the method to the TD(λ) method due to Sutton [Sut88], and to the modified policy iteration method of Puterman [Put94]. Finally, we illustrate the use of our method in the training of a tetris playing program.

The methodology of this paper also has some conceptual value. It introduces the notion of temporal differences into the mainstream DP framework, outside of the simulation-oriented context used for TD(λ) methods. The parameter λ is interpreted, for the first time, as a discount factor in the classical DP sense. As a result, the role of λ as a device for variance reduction in a simulation-based policy evaluation process is highlighted and clarified.

2. TEMPORAL DIFFERENCES-BASED POLICY ITERATION

When the number of states is large, it is usually preferable to carry out the policy evaluation step of the policy iteration method by using an iterative method such as value iteration. A drawback of this approach, however, is that the value iteration algorithm may converge very slowly. This is true for many discounted problems with a discount factor that is close to 1. We are thus motivated to use an approach whereby the discount factor is effectively reduced in order to accelerate the policy evaluation step.

The idea underlying the approach is that the discount factor can be reduced without altering the cost-to-go of a given policy only if the expected value of the one-stage cost under that policy is equal to 0. We thus introduce a transformation that asymptotically induces this one-stage cost structure. The transformation is based on the notion of *temporal differences* (TD for short), which is of major importance in the context of the simulation-based Neuro-Dynamic Programming methods (see the survey paper [BBS95], the textbook [Ber95], or the monograph [BeT96] for detailed discussions of these methods and references).

We now describe a policy iteration-like algorithm that maintains a cost function-policy pair (J_t, μ_t). We can view J_t as an approximation to the cost-to-go vector J^{μ_t}. In the typical iteration, given (J_t, μ_t), we calculate μ_{t+1} by the policy improvement step

$$T_{\mu_{t+1}} J_t = T J_t. \tag{2.1}$$

To calculate J_{t+1}, we define the TD associated with each transition (i, j) under μ_{t+1}:

$$d_t(i,j) = g\big(i, \mu_{t+1}(i), j\big) + \alpha J_t(j) - J_t(i). \tag{2.2}$$

We also consider a parameter λ from the range $[0, 1]$. We view $d_t(i, j)$ as the one-stage cost of policy μ_{t+1} for an $\alpha\lambda$-discounted DP problem with the transition probabilities $p_{ij}\big(\mu_{t+1}(i)\big)$ of the original problem, and we calculate the corresponding cost-to-go vector. This vector, denoted by Δ_t, has components given by

$$\Delta_t(i) = \sum_{m=0}^{\infty} E\big[(\alpha\lambda)^m d_t(i_m, i_{m+1}) \;\big|\; i_0 = i\big], \qquad \forall\ i. \tag{2.3}$$

The vector J_{t+1} is then obtained by

$$J_{t+1} = J_t + \Delta_t. \tag{2.4}$$

We refer to the method as the *λ-policy iteration method*.

Note that when $\lambda = 1$, by using the TD definition (2.2) in the expression (2.3), we obtain

$$\begin{aligned}
\Delta_t(i) &= \sum_{m=0}^{\infty} E\big[\alpha^m d_t(i_m, i_{m+1}) \;\big|\; i_0 = i\big] \\
&= \sum_{m=0}^{\infty} E\Big[\alpha^m g\big(i_m, \mu_{t+1}(i_m), i_{m+1}\big) + \alpha^{m+1} J_t(i_{m+1}) - \alpha^m J_t(i_m) \;\big|\; i_0 = i\Big] \\
&= \sum_{m=0}^{\infty} E\Big[\alpha^m g\big(i_m, \mu_{t+1}(i_m), i_{m+1}\big) \;\big|\; i_0 = i\Big] - J_t(i) \\
&= J^{\mu_{t+1}}(i) - J_t(i).
\end{aligned}$$

Thus, by using Eq. (2.4), we see that $J_{t+1} = J^{\mu_{t+1}}$, so *the 1-policy iteration method coincides with the standard policy iteration method*. However, for $\lambda < 1$, the two methods are different, and in fact it can be seen from Eqs. (2.2)-(2.4) that *the 0-policy iteration method coincides with the value iteration method* for the original problem.

For an alternative interpretation of the iteration (2.3) and (2.4), consider a modified policy iteration approach whereby, given μ_{t+1} and J_t, we evaluate $J^{\mu_{t+1}}$ approximately using M value iterations starting with J_t, and let $\hat{J}_{t,M}$ be the resulting function; that is,

$$\hat{J}_{t,M} = T^M J_t.$$

By using the definition (2.2) of TD, we have

$$\begin{aligned}\hat{J}_{t,M}(i) &= \alpha^M J_t(i_M) + \sum_{k=0}^{M-1} E\Big\{\alpha^k g\big(i_k, \mu_{t+1}(i_k), i_{k+1}\big) \;\Big|\; i_0 = i\Big\} \\ &= J_t(i) + \sum_{k=0}^{M-1} E\big\{\alpha^k d_t(i_k, i_{k+1}) \;\big|\; i_0 = i\big\}.\end{aligned} \tag{2.5}$$

Suppose now that λ is chosen from $[0, 1)$, and that the number M of value iterations is random and geometrically distributed with parameter λ; that is,

$$\text{Prob}(M = m) = (1 - \lambda)\lambda^{m-1}, \qquad m = 1, 2, \ldots \tag{2.6}$$

Then, using Eqs. (2.3), (2.4), (2.5), and (2.6), we have

$$\begin{aligned}E[\hat{J}_{t,M}(i)] &= (1 - \lambda) \sum_{m=1}^{\infty} \lambda^{m-1} J_{t,m}(i) \\ &= J_t(i) + (1 - \lambda) \sum_{m=1}^{\infty} \lambda^{m-1} \sum_{k=0}^{m-1} E\big\{\alpha^k d_t(i_k, i_{k+1}) \mid i_0 = i\big\} \\ &= J_t(i) + \sum_{m=0}^{\infty} E\big\{(\alpha\lambda)^m d_t(i_m, i_{m+1}) \;\big|\; i_0 = i\big\} \\ &= J_t(i) + \Delta_t(i) \\ &= J_{t+1}(i).\end{aligned}$$

Thus, J_{t+1} can be interpreted as the *expected* outcome of a modified policy iteration that starts with J_t and involves a number M of value iterations that is geometrically distributed with mean $E[M] = 1/(1 - \lambda)$. This interpretation is helpful in understanding the nature of the convergence properties of the λ-policy iteration method to be discussed.

The motivation for the λ-policy iteration method is that the $\alpha\lambda$-discounted policy evaluation step [cf. Eqs. (2.3)-(2.4)] can be much easier when $\lambda < 1$ than when $\lambda = 1$. In particular, when value iteration is used for policy evaluation, the convergence rate is faster when $\lambda < 1$ than when $\lambda = 1$. Similarly, when policy evaluation is performed using Monte-Carlo simulation, as in some of the NDP methods to be discussed in the next section, the variance of the cost samples that are averaged by simulation is typically smaller when $\lambda < 1$ than when $\lambda = 1$, as can be seen from the definition of Δ_t [the variance of the random TD terms in the summation is reduced with λ, cf. Eq. (2.3)]. As a result, the number of cost samples that are required to evaluate by simulation the cost function of a given policy within a given accuracy may be much smaller when $\lambda < 1$ than when $\lambda = 1$.

On the negative side, we will show shortly that the asymptotic convergence rate of the sequence J_t produced by the λ-policy iteration method deteriorates as λ becomes smaller. However,

this disadvantage may not be very significant within a context where cost function approximations are used, as will be discussed later (see Sections 3 and 4).

The following proposition introduces a basic mapping that underlies the λ-policy iteration method, and provides some basic results. The proposition applies to both cases where $\alpha < 1$ and $\alpha = 1$.

Proposition 2: Given $\lambda \in [0, 1)$, J_t, and μ_{t+1}, consider the mapping M_t defined by

$$M_t J = (1 - \lambda) T_{\mu_{t+1}} J_t + \lambda T_{\mu_{t+1}} J. \tag{2.7}$$

Assume that $T_{\mu_{t+1}}$ is a contraction mapping of modulus β with respect to some norm $\|\cdot\|$.

(a) The mapping M_t is a contraction mapping of modulus $\beta\lambda$ with respect to the norm $\|\cdot\|$.

(b) For any integer $m \geq 1$ and vector J, there holds

$$M_t^m J = (1 - \lambda)\big(T_{\mu_{t+1}} J_t + \lambda T_{\mu_{t+1}}^2 J_t + \cdots + \lambda^{m-1} T_{\mu_{t+1}}^m J_t\big) + \lambda^m T_{\mu_{t+1}}^m J. \tag{2.8}$$

(c) The vector J_{t+1} generated next by the λ-policy iteration method [cf. Eqs. (2.3)-(2.4)] is the unique fixed point of M_t. Furthermore,

$$J_{t+1} = (1 - \lambda) \sum_{m=0}^{\infty} \lambda^m T_{\mu_{t+1}}^{m+1} J_t. \tag{2.9}$$

Proof: (a) For any two vectors J and $\overline{J}$, using the definition (2.7) of M_t, we have

$$\|M_t J - M_t \overline{J}\| = \big\|\lambda(T_{\mu_{t+1}} J - T_{\mu_{t+1}} \overline{J})\big\| = \lambda \|T_{\mu_{t+1}} J - T_{\mu_{t+1}} \overline{J}\| \leq \beta\lambda \|J - \overline{J}\|.$$

(b) The relation (2.8) holds for $m = 1$ by the definition (2.7) of M_t. It can be proved for all $m \geq 1$ by using a straightforward induction. In particular, we have

$$M_t^{m+1} J = M_t(M_t^m J) = (1 - \lambda) T_{\mu_{t+1}} J_t + \lambda T_{\mu_{t+1}}(M_t^m J),$$

and after the expression (2.8) is used in the above equation, we obtain Eq. (2.8) with m replaced by $m + 1$.

(c) From the definition (2.3)-(2.4) of J_{t+1}, we have

$$J_{t+1} - J_t = \overline{d}_t + \alpha\lambda P_{\mu_{t+1}}(J_{t+1} - J_t), \tag{2.10}$$

where $P_{\mu_{t+1}}$ is the transition probability matrix corresponding to the policy μ_{t+1}, and $\overline{d}_t$ is the vector of expected TD, with components given by

$$\overline{d}_t(i) = \sum_j p_{ij}\big(\mu_{t+1}(i)\big) d_t(i, j), \qquad \forall\ i.$$

Using the definition (2.2) of the TD, we have

$$\overline{d}_t = T_{\mu_{t+1}} J_t - J_t, \tag{2.11}$$

so Eq. (2.10) yields

$$\begin{aligned} J_{t+1} &= T_{\mu_{t+1}} J_t + \alpha\lambda P_{\mu_{t+1}}(J_{t+1} - J_t) \\ &= T_{\mu_{t+1}} J_t + \lambda(T_{\mu_{t+1}} J_{t+1} - T_{\mu_{t+1}} J_t) \\ &= (1-\lambda) T_{\mu_{t+1}} J_t + \lambda T_{\mu_{t+1}} J_{t+1} \\ &= M_t J_{t+1}. \end{aligned}$$

Thus, J_{t+1} is the fixed point of M_t. The expression (2.9) follows from Eq. (2.8) by taking the limit as $m \to \infty$. **Q.E.D.**

The following proposition shows the validity of the λ-policy iteration method and provides its convergence rate.

Proposition 3: Assume that $\lambda \in [0, 1)$, and let (J_t, μ_t) be the sequence generated by the λ-policy iteration algorithm. Then J_t converges to J^*. Furthermore, for all t greater than some index $\overline{t}$, we have

$$\|J_{t+1} - J^*\|_\infty \leq \frac{\alpha(1-\lambda)}{1-\alpha\lambda} \|J_t - J^*\|_\infty, \tag{2.12}$$

where $\|\cdot\|_\infty$ denotes the maximum norm.

Proof: Let us first assume that $TJ_0 \leq J_0$. We show by induction that for all t, we have

$$J^* \leq TJ_{t+1} \leq J_{t+1} \leq TJ_t \leq J_t. \tag{2.13}$$

To this end, we fix t and we assume that $TJ_t \leq J_t$. We will show that $J^* \leq TJ_{t+1} \leq J_{t+1} \leq TJ_t$, and then Eq. (2.13) will follow from the hypothesis $TJ_0 \leq J_0$.

Using the fact $T_{\mu_{t+1}} J_t = TJ_t$ [cf. Eq. (2.1)] and the definition of M_t [cf. Eq. (2.7)], we have

$$M_t J_t = T_{\mu_{t+1}} J_t = TJ_t \leq J_t.$$

It follows from the monotonicity of $T_{\mu_{t+1}}$, which implies monotonicity of M_t, that for all positive integers m, we have $M_t^{m+1} J_t \leq M_t^m J_t \leq TJ_t \leq J_t$, so by taking the limit as $m \to \infty$, we obtain

$$J_{t+1} \leq TJ_t \leq J_t. \tag{2.14}$$

From the definition of M_t, we have

$$\begin{aligned} M_t J_{t+1} &= T_{\mu_{t+1}} J_t + \lambda(T_{\mu_{t+1}} J_{t+1} - T_{\mu_{t+1}} J_t) \\ &= T_{\mu_{t+1}} J_{t+1} + (1-\lambda)(T_{\mu_{t+1}} J_t - T_{\mu_{t+1}} J_{t+1}), \end{aligned}$$

Using the already shown relation $J_t - J_{t+1} \geq 0$ and the monotonicity of $T_{\mu_{t+1}}$, we obtain $T_{\mu_{t+1}} J_t - T_{\mu_{t+1}} J_{t+1} \geq 0$, so that

$$T_{\mu_{t+1}} J_{t+1} \leq M_t J_{t+1}.$$

Since $M_t J_{t+1} = J_{t+1}$, it follows that

$$TJ_{t+1} \leq T_{\mu_{t+1}} J_{t+1} \leq J_{t+1}. \tag{2.15}$$

Finally, the above relation and the monotonicity of $T_{\mu_{t+1}}$ imply that for all positive integers m, we have $T^m_{\mu_{t+1}} J_{t+1} \leq T_{\mu_{t+1}} J_{t+1}$, so by taking the limit as $m \to \infty$, we obtain

$$J^* \leq J^{\mu_{t+1}} \leq T_{\mu_{t+1}} J_{t+1}. \tag{2.16}$$

From Eqs. (2.14)-(2.16), we see that the induction proof of Eq. (2.13) is complete.

From Eq. (2.13), it follows that the sequence J_t converges to some limit $\hat{J}$ with $J^* \leq \hat{J}$. Using the definition (2.7) of M_t, and the facts $J_{t+1} = M_t J_{t+1}$ and $T_{\mu_{t+1}} J_t = TJ_t$, we have

$$J_{t+1} = M_t J_{t+1} = TJ_t + \lambda(T_{\mu_{t+1}} J_{t+1} - T_{\mu_{t+1}} J_t),$$

so by taking the limit as $t \to \infty$ and by using the fact $J_{t+1} - J_t \to 0$, we obtain $\hat{J} = T\hat{J}$. Thus $\hat{J}$ is a solution of Bellman's equation, and it follows that $\hat{J} = J^*$.

To show the result without the assumption $TJ_0 \leq J_0$, note that we can replace J_0 by a vector $\hat{J}_0 = J_0 + se$, where $e = (1, \ldots, 1)$ is the unit vector and s is a scalar that is sufficiently large so that we have $T\hat{J}_0 \leq \hat{J}_0$; it can be seen that for any scalar $s \geq (1-\alpha)^{-1} \max_i \big(TJ_0(i) - J_0(i)\big)$, the relation $T\hat{J}_0 \leq \hat{J}_0$ holds. Consider the λ-policy iteration algorithm started with $(\hat{J}_0, \mu_0)$, and let $(\hat{J}_t, \hat{\mu}_t)$ be the generated sequence. Then it can be verified by induction that for all t we have

$$\hat{J}_t - J_t = \left(\frac{\alpha(1-\lambda)}{1-\alpha\lambda}\right)^t s, \qquad \hat{\mu}_t = \mu_t.$$

Since $\hat{J}_t - J_t \to 0$ and we have already shown that $\hat{J}_t \to J^*$, it follows that $J_t \to J^*$ as well.

Since $J_t \to J^*$, it follows that for all t larger than some index $\overline{t}$, μ_{t+1} is an optimal policy, so that $T_{\mu_{t+1}} J^* = TJ^* = J^*$. By using this fact, Eq. (2.9), and the linearity of $T_{\mu_{t+1}}$, we obtain for all $t \geq \overline{t}$,

$$\begin{aligned}
\|J_{t+1} - J^*\|_\infty &= \left\| (1-\lambda) \sum_{m=0}^{\infty} \lambda^m T^{m+1}_{\mu_{t+1}} J_t - J^* \right\|_\infty \\
&= (1-\lambda) \left\| \sum_{m=0}^{\infty} \lambda^m T^{m+1}_{\mu_{t+1}} (J_t - J^*) \right\|_\infty \\
&\leq (1-\lambda) \sum_{m=0}^{\infty} \lambda^m \alpha^{m+1} \|J_t - J^*\|_\infty \\
&= \frac{\alpha(1-\lambda)}{1-\alpha\lambda} \|J_t - J^*\|_\infty.
\end{aligned}$$

Q.E.D.

Extension to Stochastic Shortest Path Problems

Let us consider the extension of our results to undiscounted problems of the stochastic shortest path type. Here, we assume that there is no discounting ($\alpha = 1$) and, to make the cost-to-go meaningful, we assume that there exists a *cost-free termination state*, denoted by 0. Once the system reaches that state, it remains there at no further cost, that is,

$$p_{00}(u) = 1, \qquad g(0, u, 0) = 0, \qquad \forall\ u \in U(0).$$

We are interested in problems where reaching the termination state is inevitable, at least under an optimal policy. Thus, the essence of the problem is how to reach the termination state with minimum expected cost.

In order to guarantee the inevitability of termination under an optimal policy, we introduce certain conditions that involve the notion of a *proper policy*; that is, a stationary policy that leads to the termination state with probability one, regardless of the initial state.

A stationary policy μ is said to be *proper* if, using this policy, there is positive probability that the termination state will be reached after at most n stages, regardless of the initial state, that is, if

$$\rho_\mu = \max_{i=1,\ldots,n} \mathrm{P}\big(i_n \neq 0 \mid i_0 = i, \mu\big) < 1. \tag{2.17}$$

A stationary policy that is not proper is said to be *improper*.

Throughout this paper and whenever we are dealing with stochastic shortest path problems, we assume the following.

Assumption 1: There exists at least one proper policy. Furthermore, for every improper policy μ, the corresponding cost-to-go $J^\mu(i)$ is infinite for at least one state i.

The theory of stochastic shortest path problems is developed in [BeT89] and [BeT91] (see [Ber95] and [BeT96] for detailed treatments). It is shown in these references that the results of Prop. 1 and the method of policy iteration have direct counterparts for stochastic shortest path problems under Assumption 1.

The convergence and convergence rate of Prop. 3 can be generalized so that it applies to the stochastic shortest path case. In particular, the proof of Eq. (2.12) can be adapted to show that J_t converges to J^* under Assumption 1 and the additional assumption $TJ_0 \leq J_0$. Furthermore, if $\|\cdot\|$ is a norm with respect to which T_{μ^*} is a contraction of modulus β for all optimal policies μ^*, we have

$$\|J_{t+1} - J^*\| \leq \frac{\beta(1-\lambda)}{1-\beta\lambda}\|J_t - J^*\|$$

for all sufficiently large t. In the case of a stochastic shortest path problem where all policies are proper, the above relation holds with $\|\cdot\|$ being a weighted maximum norm with respect to which T and all T_μ are contractions (see [BeT96], Section 2.2).

3. APPROXIMATE AND OPTIMISTIC λ-POLICY ITERATION

The method of this paper was motivated by large-scale problems where simulation-based approximations of the cost-to-go function become interesting, i.e., the neuro-dynamic programming (NDP for short) or reinforcement learning context (see e.g., [BBS95], [BeT96]). When policy evaluation is performed using Monte-Carlo simulation, the variance of the cost samples that are averaged by simulation is typically smaller when $\lambda < 1$ than when $\lambda = 1$, as can be seen from the definition of Δ_t [cf. Eq. (2.3)]. As a result, the number of cost samples that are required to evaluate by simulation the cost-to-go of a given policy within a given accuracy may be much smaller when $\lambda < 1$ than when $\lambda = 1$. On the other hand, the disadvantage of the slower convergence rate for smaller values of λ may not be very significant. The reason is that when the policy evaluation step cannot be performed with high accuracy, a reasonable practical objective is to obtain a policy whose cost is within the best "achievable" tolerance from the optimum, rather than to obtain the optimal cost-to-go function and an optimal policy. Under these circumstances, the asymptotic rate of convergence of J_t may not be crucial, and using $\lambda < 1$ often requires a comparable number of policy iterations to attain the same performance level as using $\lambda = 1$. For instance, this happens in the tetris example that we discuss in Section 4.

An approximate version of the λ-policy iteration method for a stochastic shortest path problem works as follows. We introduce a linear approximation architecture of the general form

$$\tilde{J}(i,r) = r(0) + \sum_{k=1}^{K} r(k)\phi_k(i), \tag{3.1}$$

where $r(k)$, $k = 0, 1, \ldots, K$, are the components of the parameter vector r, and ϕ_k are fixed, easily computable basis functions. Each value of r provides an approximation $\tilde{J}(i,r)$ of the optimal cost-to-go function. Let r_t be the parameter vector after t policy updates, and let μ_t be the greedy policy with respect to r_t; that is,

$$\mu_t(i) = \arg\min_{u \in U(i)} \sum_{j=0}^{n} p_{ij}(u)\big(g(i,u,j) + \tilde{J}(j,r_t)\big), \qquad \forall\ i.$$

We then simulate a batch of M trajectories, and the corresponding parameter vector r_{t+1} is obtained as

$$r_{t+1} = \arg\min_r \sum_{m=1}^{M} \sum_{k=0}^{N_m} \Big(\tilde{J}(i_{m,k}, r) - \tilde{J}(i_{m,k}, r_t) - \sum_{s=k}^{N_m-1} \lambda^{s-k} d_t(i_{m,s}, i_{m,s+1}) \Big)^2, \tag{3.2}$$

where $(i_{m,0}, i_{m,1}, \ldots, i_{m,N_m-1}, i_{m,N_m})$ is the sequence of states comprising the mth trajectory in the batch, with i_{m,N_m} being equal to the termination state, and

$$d_t(i_{m,s}, i_{m,s+1}) = g\big(i_{m,s}, \mu_t(i_{m,s}), i_{m,s+1}\big) + \tilde{J}(i_{m,s+1}, r_t) - \tilde{J}(i_{m,s}, r_t)$$

are the corresponding temporal differences with $\tilde{J}(i_{m,N_m}, r_t)$ being equal to the terminal cost 0.

One may also consider optimistic variants of the λ-policy iteration method, where a parameter vector $\hat{r}_t$ is calculated by solving the minimization problem in Eq. (3.2), with the number M of games in the batch being relatively small. The new parameter vector r_{t+1} is then computed by interpolation between r_t and $\hat{r}_t$, i.e.,

$$r_{t+1} = r_t + \gamma_t(\hat{r}_t - r_t), \tag{3.3}$$

where γ_t is a stepsize that satisfies $0 < \gamma_t \leq 1$ and is diminishing with t. These variants may be viewed as incremental algorithms that resemble on-line versions of TD(λ). We note that there are similar variants of the optimistic λ-policy iteration method with varying degrees of incrementalism, which use multiple simulated trajectories at each iteration, and there are also discounted cost variants.

Consider now the case where only one trajectory $(i_0, i_1, \ldots, i_{N-1})$ is generated per policy update ($M = 1$). Given r_t, suppose that the least squares minimization (3.2) is performed approximately using a single gradient iteration. Then we have

$$r_{t+1} = r_t + \gamma_t \sum_{k=0}^{N-1} \nabla \tilde{J}(i_k, r_t) \sum_{s=k}^{N-1} \lambda^{s-k} d_t(i_s, i_{s+1}), \tag{3.4}$$

where γ_t is the stepsize. It can also be verified that Eq. (3.4) is identical to the TD(λ) iteration of Sutton [Sut88]. Thus, we can view TD(λ) as an approximate version of the λ-policy iteration method where the least squares minimization (3.2) is approximated using a single trajectory, and a single gradient iteration.

We believe that because TD(λ) is subject to the potential sources of unreliability of gradient-like methods (ill-conditioning, difficulty with choosing the stepsize γ_t), it may be inherently less

robust than the approximate λ-policy iteration (3.2), which can be performed using efficient and numerically stable linear algebra packages.

On the other hand, if a nonlinear approximation architecture is used in place of Eq. (3.1), one must solve a nonlinear least squares problem in Eq. (3.2), and much of the potential advantage over TD(λ) is lost. We also mention one more theoretical advantage of TD(λ). It is possible to show that if the architecture is linear, the policy is fixed, and some additional technical conditions are satisfied, TD(λ) is a convergent algorithm (see [BeT96], Section 6.3). By contrast, no comparable result has been shown for the iteration (3.2), even when the policy is kept fixed.

4. APPLICATION TO TETRIS

Tetris is a popular video game played on a two-dimensional grid. Each square in the grid can be full or empty, making up a "wall of bricks" with "holes." The squares fill up as objects of different shapes fall from the top of the grid and are added to the top of the wall, giving rise to a "jagged top." Each falling object can be moved horizontally and can be rotated by the player in all possible ways, subject to the constraints imposed by the sides of the grid. There is a finite set of standard shapes for the falling objects. The game starts with an empty grid and ends when a square in the top row becomes full and the top of the wall reaches the top of the grid. However, when a row of full squares is created, this row is removed, the bricks lying above this row move one row downward, and the player scores a point. (More than one row can be created and removed in a single step, in which case the number of points scored by the player is equal to the number of rows removed.) The player's objective is to maximize the score attained (total number of rows removed) up to termination of the game.

It has been shown that for every policy the game terminates with probability 1 (see [Bur97]), so we can model the problem of finding an optimal tetris playing strategy as a stochastic shortest path problem, where Assumption 1 is satisfied. The control, denoted by u, is the horizontal positioning and rotation applied to the falling object. The state consists of two components:

(1) The board position, that is, a binary description of the full/empty status of each square, denoted by i.

(2) The shape of the current falling object, denoted by y.

As soon as the most recent object has been placed, the new component y is generated

according to a probability distribution $p(y)$, independently of the preceding history of the game. Therefore (as shown in Example 2.2 of [BeT96]), it is possible to use the reduced form of Bellman's equation involving a reward-to-go vector $\hat{J}$ that depends only on the component i of the state. This equation has the form

$$\hat{J}(i) = \sum_y p(y) \max_u \Big[g(i,y,u) + \hat{J}\big(f(i,y,u)\big) \Big], \qquad \forall\ i,$$

where $g(i,y,u)$ and $f(i,y,u)$ are the number of points scored (rows removed), and the next board position, respectively, when the state is (i,y) and control u is applied.

Unfortunately, the number of states in the tetris problem is extremely large. It is roughly equal to $m2^{hw}$, where m is the number of different shapes of falling objects, and h and w are the height and width of the grid, respectively. In particular, for the reasonable numbers $m = 7$, $h = 20$, and $w = 10$ we have over 10^{61} states. Thus it is essential to use approximations. The control selection methods of NDP use a scoring function, which can be viewed as an approximation to the optimal reward function. In particular, the NDP methods that we will discuss later, construct scoring functions that evaluate a tetris position based on some characteristic features of the position. Such features are easily recognizable by experienced players, and include the current height of the wall, the presence of "holes" and "glitches" (severe irregularities) in the first few rows, etc.

We have trained a linear feature-based approximation architecture using approximate and optimistic λ-policy iteration. After some experimentation, the following features were used:

(a) The height h_k of the kth column of the wall. There are w such features, where w is the wall's width.

(b) The absolute difference $|h_k - h_{k+1}|$ between the heights of the kth and the $(k+1)$st column, $k = 1, \ldots, w-1$.

(c) The maximum wall height $\max_k h_k$.

(d) The number of holes L in the wall, that is, the number of empty positions of the wall that are surrounded by full positions.

Thus, there are $2w+1$ features, which together with a constant offset, require $2w+2$ weights in a linear architecture of the form

$$\begin{aligned}\tilde{J}(i,r) = r(0) + \sum_{k=1}^{w} r(k) h_k + \sum_{k=1}^{w-1} r(k+w) |h_k - h_{k+1}| \\ + r(2w) \max_k h_k + r(2w+1) L.\end{aligned}$$

We tried an approximate version of the λ-policy iteration method. In particular, let r_t be the weight vector after t iterations, and let μ_t be the greedy policy with respect to r_t. The policy μ_t is evaluated using a batch of M (in the order of 100) games, and the corresponding weight vector r_{t+1} is obtained as

$$r_{t+1} = \arg\min_{r} \sum_{m=1}^{M} \sum_{k=0}^{N_m} \Big(\tilde{J}(i_{m,k}, r) - \tilde{J}(i_{m,k}, r_t) - \sum_{s=k}^{N_m-1} \lambda^{s-k} d(i_{m,s}, i_{m,s+1}) \Big)^2, \tag{4.1}$$

where $(i_{m,0}, i_{m,1}, \ldots, i_{m,N_m-1}, i_{m,N_m})$ is the sequence of states comprising the mth game in the batch, with i_{m,N_m} being equal to the termination state, and

$$d(i_{m,s}, i_{m,s+1}) = g\big(i_{m,s}, \mu_t(i_{m,s}), i_{m,s+1}\big) + \tilde{J}(i_{m,s+1}, r_t) - \tilde{J}(i_{m,s}, r_t)$$

are the corresponding temporal differences with $\tilde{J}(i_{m,N_m}, r_t)$ being equal to the terminal value 0. All games were started from the empty board position (things did not change much when the initial board position was chosen more randomly).

We also tried an optimistic version of the λ-policy iteration method, whereby a weight vector $\hat{r}_t$ was calculated by solving the minimization problem in Eq. (4.1), with the number M of games in the batch being relatively small (in the order of 5). The new weight vector r_{t+1} was then computed by interpolation between r_t and $\hat{r}_t$, i.e.,

$$r_{t+1} = r_t + \gamma_t(\hat{r}_t - r_t), \tag{4.2}$$

where γ_t is a stepsize that satisfies $0 < \gamma_t \leq 1$ and is diminishing with t.

We now describe some of the results of the computational experimentation. The wall width w was taken to be 10, the wall height was taken to be 20, and the types of falling objects were the 7 possible shapes that consist of 4 pieces. Each falling object was chosen with equal probability from the 7 possible shapes, independently of the shapes of the preceding objects. The starting set of weights in our experiments was $r(2w) = 10$, $r(2w+1) = 1$, and $r(k) = 0$ for $k < 2w$ (this set of weights was derived from those used by Van Roy [Van95]). With the weights fixed at these initial values, the corresponding greedy policy scores in the low tens.

The approximate λ-policy iteration method quickly gave playing policies that score in the thousands, except when $\lambda = 1$, in which case the method failed to make satisfactory progress. We attribute the failure for $\lambda = 1$ to the high variance of the game scores. Generally, the maximum score achieved depended on the value of λ. Table 1 gives some illustrative results with different values of λ. Figure 1 shows the sequence of tetris scores obtained during training for the case

where $\lambda = 0$, $\lambda = 0.3$, $\lambda = 0.5$, and $\lambda = 0.7$. An interesting and somewhat paradoxical observation is that a high performance is achieved after relatively few policy iterations, but the performance gradually drops significantly. We have no explanation for this intriguing phenomenon, which occurred with all of the successful methods that we tried. In particular, the weights corresponding to the high scoring policies did not have any distinguising characteristics relative to the weights corresponding to the policies obtained at convergence.

λ	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Score	2909	2818	2730	2968	3014	2786	3183	1941	2103	1054

Table 1: Average scores of the highest scoring policies obtained using different values of λ after 15 policy/weight updates, with 100 games between policy updates. Each data point is the average of the scores of 100 games.

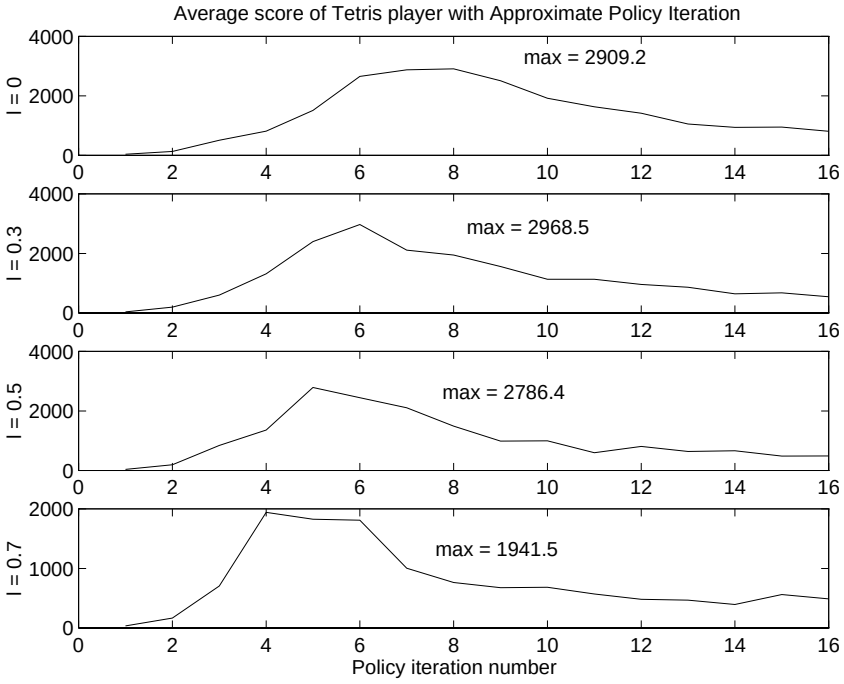

Figure 1: Sequence of tetris scores of the policies generated using approximate λ-policy iteration, for four different values of λ. Each data point is the average of the scores of 100 games.

The optimistic version of the method produced similar results to the nonoptimistic version. Figure 2 shows the sequence of tetris scores obtained during training for the case where 5 games were used per policy iteration and $\lambda = 0.6$. The stepsize used was of the form $a/(b+t)$, where t is

the policy index. The proper value of a strongly depends on λ. In particular, smaller values of a are required for λ close to 1, since the weight update increments tend to be larger as λ increases. On the whole, however, it was not difficult to obtain reasonable stepsize parameter values by trial and error.

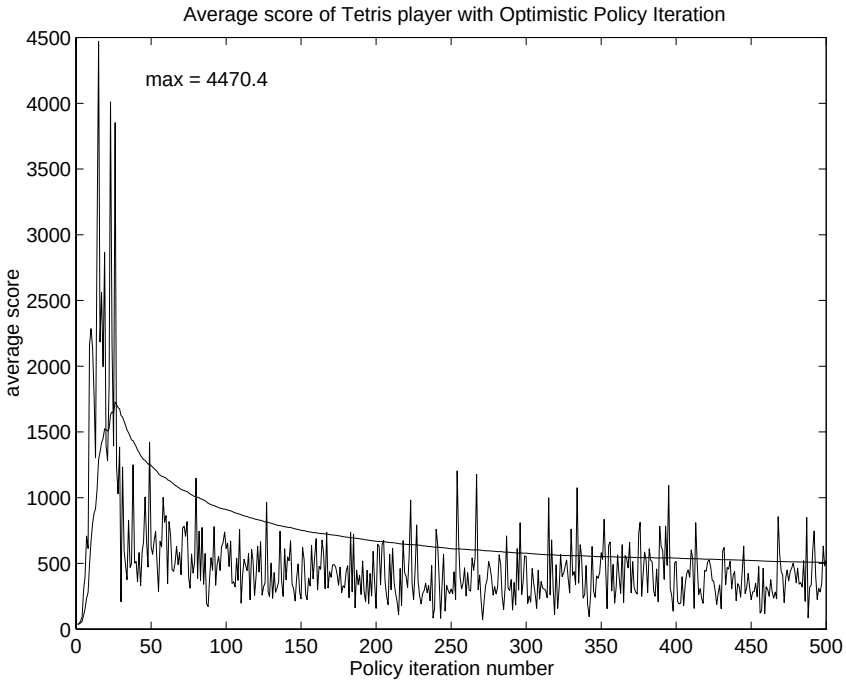

Figure 2: Sequence of tetris scores of the policies generated using optimistic λ-policy iteration, where $\lambda = 0.6$. Each data point is the average of the scores of 5 games. The cumulative average score is also shown.

We finally mention two additional approaches that were used for training tetris players. The first was a policy iteration approach, where the weights of the architecture were updated using the TD(λ) method of Sutton [Sut88]. In this approach we kept the policy fixed for a substantial number of games (of the order of a 100), during which we evaluated approximately the cost-to-go of the policy using TD(λ). The policy evaluation, once completed, was used to define an "improved" policy, in the spirit of policy iteration. We used the same linear architecture as for the λ-policy iteration method. For a fixed policy, it is possible to show that the implementation of the TD(λ) method that we used is convergent for our problem (see [BeT96], Section 6.3.4). Despite this fact, the TD(λ) approach ran into serious difficulties, typically failing to make substantial progress, and was abandoned in favor of the much better performing λ-policy iteration method. This failure can probably be attributed to difficulties with large simulation noise and/or the ill-conditioning to which gradient-like methods such as TD(λ) are susceptible.

In the second approach, the problem was altered by imposing a cost structure that provides an incentive for not losing quickly rather than for achieving a high score. In particular, the objective was reformulated so that maximization of the average game score, was replaced by minimization of the total number of wall height increases in the course of placing the next N falling objects, where N is a fixed integer (values between 10 and 100 were used). Thus, for a given stage where the maximum wall height changes from a value of h to a value of $\overline{h}$, the cost is $\max\{0, \overline{h} - h\}$. Also, to discourage termination, a fixed terminal cost G was introduced (values between 5 and 20 were used). Thus the problem was transformed to a stochastic shortest path problem, where the termination occurs in at most N stages. The advantage of this formulation is that the variance of the cost samples is significantly reduced. The two problem formulations are substantially different, but it was reasoned that a good policy under one formulation should also be good for the other. Note that it is essential to fix the number of stages within which to count height increases, because otherwise a good tetris playing policy, which achieves a high score, would perform very poorly in terms of number of height increases over a long horizon.

Based on the alternative truncated horizon problem formulation just described, an approximate and an optimistic version of the λ-policy iteration method of Section 3 were tried. The results obtained were similar but somewhat more favorable than the ones presented in Table 1. This occurred uniformly for all values of λ. The method also worked for $\lambda = 1$ and attained a score of 1015 using 100 games between policy updates, and 1637 using 300 games between policy updates. A maximum score of 3554 was obtained for $\lambda = 0.3$ (using 100 games between policy updates), which should be compared with the best score of 3183 given in Table 1 ($\lambda = 0.6$). We were again unable to get TD(λ) to work for the truncated horizon problem formulation.

5. REFERENCES

[BBS95] Barto, A. G., Bradtke, S. J., and Singh, S. P., 1995. "Learning to Act Using Real-Time Dynamic Programming," Artificial Intelligence, Vol. 72, pp. 81-138.

[BeT89] Bertsekas, D. P., and Tsitsiklis, J. N., 1989. Parallel and Distributed Computation: Numerical Methods, Prentice-Hall, Englewood Cliffs, N. J.

[BeT91] Bertsekas, D. P., and Tsitsiklis, J. N., 1991. "An Analysis of Stochastic Shortest Path Problems," Mathematics of Operations Research, Vol. 16, pp. 580-595.

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

[Ber95] Bertsekas, D. P., 1995. Dynamic Programming and Optimal Control, Vols. I and II, Athena Scientific, Belmont, MA.

[Bur97] Burgiel, H., 1997. "How to Lose in Tetris," Preprint, The Geometry Center, Minneapolis, MN.

[Put94] Puterman, M. L., 1994. Markovian Decision Problems, Wiley, N. Y.

[Ros83] Ross, S. M., 1983. Introduction to Stochastic Dynamic Programming, Academic Press, N. Y.

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

[Van95] Van Roy, B., 1995. "Feature-Based Methods for Large Scale Dynamic Programming," Lab. for Info. and Decision Systems Report LIDS-TH-2289, Massachusetts Institute of Technology, Cambridge, MA.