CRICOS
00213J

ENN582: Reinforcement Learning and Optimal Control

Dynamic Programming for Infinite Horizon Problems

Daniel E. Quevedo[‡]

VERSION 1.

Abstract

The purpose of these notes is to introduce infinite horizon optimal control problems and present basic algorithms for their solution. More details can be found in books, such as [6, 1, 5].

1 Key Learning Objectives

In conjunction with the practical, you will learn to

- formulate infinite horizon stochastic control problems
- understand the difference between value iteration and policy iteration algorithms
- implement value iteration and policy iteration algorithms.

[‡]Please report errors within this document to daniel.quevedo@qut.edu.au

2 Infinite Horizon Formulation

We recall that (time-invariant) stochastic systems can be modelled as per

$$x_{k+1} = f(x_k, u_k, w_k), \quad k \in \mathbb{N}_0, \quad x_k \in \mathbb{X},\ u_k \in \mathbb{U},\ w_k \in \mathbb{W}, \quad x_0 = x, \tag{1}$$

which, for discrete-space models, can equivalently be described in terms of the transition probabilities

$$p_{ij}(u) = \text{Prob}(x_{k+1} = j | x_k = i, u_k = u), \quad i, j \in \mathbb{X},\ u_k \in \mathbb{U}. \tag{2}$$

So far we have studied control problems over a finite horizon. These are, at times, called "episodic tasks". To be able to tackle situations where systems operate over a very long time ("continuing tasks"), we can use the receding horizon optimisation principle. Alternatively, we can directly state optimisation problems over an infinite horizon. Whilst, at first sight, one might expect that the computations required to compute optimal control policies becomes prohibitive, it turns out that this is at times not the case. In fact, for time-invariant systems of the form (1), and if we consider problems with a common cost function for all stages, then infinite horizon problems typically lead to stationary control policies.

We will focus on <u>discounted</u> infinite horizon problems with a common cost function for all stages:

$$J_{\pi^*}(x) = \min_{\pi \in \Pi} J_\pi(x), \quad J_\pi(x) = \lim_{N\to\infty} \underset{w_k}{E} \left\{ \sum_{k=0}^{N-1} \alpha^k g(x_k, \mu_k(x_k), w_k) \right\}, \quad x_0 = x, \tag{3}$$

where g is the stage cost and Π is the set of all admissible policies $\pi = \{\mu_0, \mu_1, \dots\}$ with $\mu_k \colon \mathbb{X} \to \mathbb{U}$.[1]

In (3), $\alpha \in [0, 1)$ is the "discount factor" (or "discount rate"). The value of α chosen represents the present importance of future costs. Costs incurred in the future are discounted. The extreme case of $\alpha = 0$ describes a myopic scenario, wherein one is only interested in the current cost. If α is chosen closer to one, then the control problem puts more emphasis on the effect of current decisions on future states and, thereby, future actions and costs.

Before proceeding to derive algorithms that recursively find solutions to (3), we first note that

$$\begin{aligned} J_\pi(x_0) &= \underset{w_k}{E}\left\{ g(x_0, \mu_0(x_0), w_0) + \alpha g(x_1, \mu_1(x_1), w_1) + \alpha^2 g(x_2, \mu_2(x_2), w_2) + \dots \right\} \\ &= \underset{w_0}{E}\big\{g(x_0, \mu_0(x_0), w_0)\big\} + \underset{w_k}{E}\left\{ \alpha g(x_1, \mu_1(x_1), w_1) + \alpha^2 g(x_2, \mu_2(x_2), w_2) + \dots \right\} \\ &= \underset{w_0}{E}\big\{g(x_0, \mu_0(x_0), w_0)\big\} + \underset{w_k}{E}\left\{ \alpha\Big(g(x_1, \mu_1(x_1), w_1) + \alpha g(x_2, \mu_2(x_2), w_2) + \dots \Big) \right\} \\ &= \underset{w_0}{E}\big\{g(x_0, \mu_0(x_0), w_0)\big\} + \alpha \underset{w_k}{E}\left\{ g(x_1, \mu_1(x_1), w_1) + \alpha g(x_2, \mu_2(x_2), w_2) + \dots \right\} \\ &= \underset{w_0}{E}\big\{g(x_0, \mu_0(x_0), w_0)\big\} + \alpha J_\pi(x_1). \end{aligned} \tag{4}$$

[1]To avoid mathematical difficulties, we shall assume that, for every finite value $x \in \mathbb{X}$, the optimal cost $J_{\pi^*}(x)$ is finite.

Thus, costs at different time steps are related to each other. This feature simplifies calculations and underlies some of the algorithms to be studied in the sequel.

3 Value Iteration

To elucidate the infinite horizon problem (3), let us fix the horizon N and consider an arbitrary bounded function $J\colon \mathbb{X} \to \mathbb{U}$. Accumulation of the stage cost over N stages and adding some terminal cost of the form $\alpha^N J(x_N)$ leads to the total expected cost, cf. (3):

$$\underset{w_k}{E}\left\{\alpha^N J(x_N) + \sum_{k=0}^{N-1} \alpha^k g(x_k, \mu_k(x_k), w_k)\right\}, \quad x_0 = x. \tag{5}$$

Application of the DP algorithm for stochastic systems for this time-invariant case then leads to:

$$J_{N-k}(x) = \min_{u\in\mathbb{U}} \underset{w}{E}\Big\{\alpha^{N-k} g(x,u,w) + J_{N-k+1}(f(x,u,w))\Big\}, \quad k = 1, \ldots, N \tag{6}$$

with initial condition

$$J_N(x) = \alpha^N J(x). \tag{7}$$

As before, $J_{N-k}(x)$ denotes the optimal cost to go of the last k stages starting from state x. The final step of the recursion leads to $J_0(x)$, the optimal N-stage cost.

In order to obtain an algorithm which runs forward in time, it is convenient to reverse the time index by introducing the functions

$$V_k(x) = \frac{J_{N-k}(x)}{\alpha^{N-k}}. \tag{8}$$

Note that $V_N(x) = J_0(x)$, the optimal N-stage cost. In terms of V_k, the DP recursions in (6) become:[2]

$$\alpha^{N-k} V_k(x) = \min_{u\in\mathbb{U}} \underset{w}{E}\Big\{\alpha^{N-k} g(x,u,w) + \alpha^{N-k+1} V_{k-1}\big(f(x,u,w)\big)\Big\}, \quad k = 1, \ldots, N,$$

or, equivalently,

$$V_{k+1}(x) = \min_{u\in\mathbb{U}} \underset{w}{E}\big\{g(x,u,w) + \alpha V_k\big(f(x,u,w)\big)\big\}, \quad k = 0, 1, \ldots, N-1, \quad \forall x \in \mathbb{X} \tag{9}$$

with initial condition $V_0(x) = J_N(x)/\alpha^N = J(x)$, see also (7).

The recursion (9) is called the **Value Iteration** algorithm, or VI for short. VI operates forward in time and allows us to calculate the optimal N-stage cost function $V_N(x)$ given in Equation (5) directly from the optimal $(N-1)$-stage cost function V_{N-1} as per:

$$V_N(x) = \min_{u\in\mathbb{U}} \underset{w}{E}\big\{g(x,u,w) + \alpha V_{N-1}\big(f(x,u,w)\big)\big\}, \quad \forall x \in \mathbb{X}.$$

[2] $J_{N-k+1}(f(x,u,w)) = J_{N-(k-1)}(f(x,u,w)) = \alpha^{N-(k-1)} V_{k-1}(f(x,u,w))$

Starting from some bounded function $J(\cdot)$ (which could be the zero function as well), with each iteration, we will obtain the solution to an optimisation problem with a horizon which is one stage longer than the preceding problem.

Whilst formulated for the finite horizon problem (5), VI will typically approximate the infinite horizon optimisation problem (3). Indeed, if $\underset{w}{E}\big\{g(x,u,w)\big\}$ is bounded for all $x \in \mathbb{X}$ and $u \in \mathbb{U}$, then, as we shall see below, we have convergence:

$$\lim_{N\to\infty} V_N(x) = J_{\pi^*}(x), \quad \forall x \in \mathbb{X}. \tag{10}$$

Thus, VI finds the optimal value function, and the optimal policy can be extracted as the minimiser of this cost, see (9).

The above allows us to conclude that the optimal infinite horizon policy $J_{\pi^*}(x)$ is a fixed point for (9), i.e., it satisfies **Bellman's Equation**:

$$J_{\pi^*}(x) = \min_{u\in\mathbb{U}} \underset{w}{E}\big\{g(x,u,w) + \alpha J_{\pi^*}\big(f(x,u,w)\big)\big\}, \quad \forall x \in \mathbb{X}. \tag{11}$$

Bellman Operators and Convergence of VI

It is convenient to introduce a shorthand notation for the VI recursion in (9) by considering it as a mapping, say T, which transforms a function $J\colon \mathbb{X} \to \mathbb{R}$ to another function $TJ\colon \mathbb{X} \to \mathbb{R}$, where:

$$TJ(x) = \min_{u\in\mathbb{U}} \underset{w}{E}\Big\{g(x,u,w) + \alpha J(f(x,u,w))\Big\}, \quad \forall x \in \mathbb{X}. \tag{12}$$

VI recursions then correspond to compositions of T, here denoted as:

$$(T^k J)(x) = \big(T(T^{k-1}J)\big)(x), \quad \forall x \in \mathbb{X}$$

with $(T^0 J)(x) = J(x)$.

In terms of this Bellman operator T, convergence of VI becomes:

$$J_{\pi^*}(x) = \lim_{k\to\infty} (T^k J)(x), \quad \forall x \in \mathbb{X}$$

and can be formally proven for problem instances where the expected values of the stage costs are uniformly bounded as follows:

For every positive integer K, initial state $x \in \mathbb{X}$ and policy $\pi = \{\mu_0, \mu_1, \dots\}$, we have:

$$\begin{aligned} J_\pi(x) &= \lim_{N\to\infty} \underset{w_k}{E}\left\{\sum_{k=0}^{N-1} \alpha^k g(x_k, \mu_k(x_k), w_k)\right\} \\ &= \underset{w_k}{E}\left\{\sum_{k=0}^{K-1} \alpha^k g(x_k, \mu_k(x_k), w_k)\right\} + \lim_{N\to\infty} \underset{w_k}{E}\left\{\sum_{k=K}^{N-1} \alpha^k g(x_k, \mu_k(x_k), w_k)\right\}, \end{aligned}$$

i.e.,

$$\underset{w_k}{E}\left\{\sum_{k=0}^{K-1}\alpha^k g(x_k,\mu_k(x_k),w_k)\right\} = J_\pi(x) - \lim_{N\to\infty}\underset{w_k}{E}\left\{\sum_{k=K}^{N-1}\alpha^k g(x_k,\mu_k(x_k),w_k)\right\}. \tag{13}$$

If we assume that $\underset{w}{E}|g(x,\mu_k(x),w)| \leq M$, for all x and k, then it holds that

$$\left|\lim_{N\to\infty}\underset{w_k}{E}\left\{\sum_{k=K}^{N-1}\alpha^k g(x_k,\mu_k(x_k),w_k)\right\}\right| \leq M\sum_{k=K}^{\infty}\alpha^k = \frac{\alpha^K M}{1-\alpha}.$$

and (13) provides

$$J_\pi(x) - \frac{\alpha^K M}{1-\alpha} \leq \underset{w_k}{E}\left\{\sum_{k=0}^{K-1}\alpha^k g(x_k,\mu_k(x_k),w_k)\right\} \leq J_\pi(x) + \frac{\alpha^K M}{1-\alpha}$$

so that:

$$\begin{aligned} J_\pi(x) - \frac{\alpha^K M}{1-\alpha} + \alpha^K \min_{x\in\mathbb{X}} J(x) \leq \underset{w_k}{E}&\left\{\alpha^K J(x_K) + \sum_{k=0}^{K-1}\alpha^k g(x_k,\mu_k(x_k),w_k)\right\} \\ &\leq J_\pi(x) + \frac{\alpha^K M}{1-\alpha} + \alpha^K \max_{x\in\mathbb{X}} J(x). \end{aligned}$$

By taking the minimum over all policies, the sandwiched term above corresponds to the K times composition of the VI operator T (see (12) and (5)):

$$J_{\pi^*}(x) - \frac{\alpha^K M}{1-\alpha} + \alpha^K \min_{x\in\mathbb{X}} J(x) \leq (T^K J)(x) \leq J_{\pi^*}(x) + \frac{\alpha^K M}{1-\alpha} + \alpha^K \max_{x\in\mathbb{X}} J(x).$$

The result follows by taking the limit as $K \to \infty$.

Finite Spaces

In scenarios where the state space $\mathbb{X}$, the action space $\mathbb{U}$ and the noise space $\mathbb{W}$ are discrete and finite, the state at the next time step is characterised by a transition probability:

$$p_{ij}(u) = \text{Prob}(x_{k+1} = j | x_k = i, u_k = u), \quad i,j \in \mathbb{X},\ u_k \in \mathbb{U}.$$

If we denote $g(i,u)$ as the expected cost per stage $x_k = i$ when the control u is applied, then the VI recursion (9) becomes:

$$V_{k+1}(i) = \min_{u\in\mathbb{U}}\left\{g(i,u) + \alpha\sum_{j\in\mathbb{X}} p_{ij}(u)V_k(j)\right\}, \quad i \in \mathbb{X}. \tag{14}$$

As we shall see in the following example, for finite state and action spaces, VI recursions can be restated as simple operations on matrices.

Revisiting the rover

We recall the rover example where the state space $\mathbb{X} = \{\mathrm{T}, \mathrm{R}, \mathrm{B}\}$ describes the positions "Top", "Rolling" and "Bottom," respectively. The control set is $\mathbb{U} = \{0, 1\}$, where 0 refers to "not driving" and 1 to"driving."

The dynamics of the rover is described by the transition probabilities $P(u)$ as follows (see also [4]):

$$
\begin{aligned}
P(0) &= \begin{bmatrix} p_{\mathrm{TT}}(0) & p_{\mathrm{TR}}(0) & p_{\mathrm{TB}}(0) \\ p_{\mathrm{RT}}(0) & p_{\mathrm{RR}}(0) & p_{\mathrm{RB}}(0) \\ p_{\mathrm{BT}}(0) & p_{\mathrm{BR}}(0) & p_{\mathrm{BB}}(0) \end{bmatrix} = \begin{bmatrix} 0.75 & 0.25 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \\
P(1) &= \begin{bmatrix} p_{\mathrm{TT}}(1) & p_{\mathrm{TR}}(1) & p_{\mathrm{TB}}(1) \\ p_{\mathrm{RT}}(1) & p_{\mathrm{RR}}(1) & p_{\mathrm{RB}}(1) \\ p_{\mathrm{BT}}(1) & p_{\mathrm{BR}}(1) & p_{\mathrm{BB}}(1) \end{bmatrix} = \begin{bmatrix} 0.8 & 0.2 & 0 \\ 0.9 & 0 & 0.1 \\ 0 & 0.1 & 0.9 \end{bmatrix}.
\end{aligned}
\tag{15}
$$

The stage costs are kept as before, namely

$$
g(\mathrm{T}, 0) = -3, \quad g(\mathrm{T}, 1) = -1, \quad g(\mathrm{R}, 0) = 0, \quad g(\mathrm{R}, 1) = 2, \quad g(\mathrm{B}, 0) = 0, \quad g(\mathrm{B}, 1) = 2.
$$

Control design then amounts to choosing the discount factor α.

We initialise VI with the simple choice $V_0(i) = J(i) = 0$, for all i. Expression (14) then gives:

$$
V_1(i) = \min\Big\{g(i,0) + \alpha \sum_{j\in\mathbb{X}} p_{ij}(0)V_0(j), g(i,1) + \alpha \sum_{j\in\mathbb{X}} p_{ij}(1)V_0(j)\Big\} = \min\big\{g(i,0), g(i,1)\big\},
$$

thus:

$$
V_1 = \begin{bmatrix} V_1(\mathrm{T}) \\ V_1(\mathrm{R}) \\ V_1(\mathrm{B}) \end{bmatrix} = \begin{bmatrix} \min\big\{g(\mathrm{T},0), g(\mathrm{T},1)\big\} \\ \min\big\{g(\mathrm{R},0), g(\mathrm{R},1)\big\} \\ \min\big\{g(\mathrm{B},0), g(\mathrm{B},1)\big\} \end{bmatrix} = \begin{bmatrix} \min\{-3,-1\} \\ \min\{0,2\} \\ \min\{0,2\} \end{bmatrix} = \begin{bmatrix} -3 \\ 0 \\ 0 \end{bmatrix}.
$$

More generally, it is convenient to define

$$
G(0) = \begin{bmatrix} g(\mathrm{T},0) \\ g(\mathrm{R},0) \\ g(\mathrm{B},0) \end{bmatrix} = \begin{bmatrix} -3 \\ 0 \\ 0 \end{bmatrix}, \quad G(1) = \begin{bmatrix} g(\mathrm{T},1) \\ g(\mathrm{R},1) \\ g(\mathrm{B},1) \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}
$$

so that

$$
V_{k+1} = \min\big\{G(0) + \alpha P(0)V_k, G(1) + \alpha P(1)V_k\big\}
$$

where the minimisation operates componentwise.

Figure 1 illustrates how VI converges to the optimal cost and policy. We note that the policies converge before the value functions converge. Whilst this observation seems encouraging from a computational viewpoint, unfortunately we cannot **ensure** the convergence of policies before value functions have converged.

With $\alpha = 0.9$ the policy obtained is given by

$$
\mu(\mathrm{T}) = 0, \quad \mu(\mathrm{R}) = 1, \quad \mu(\mathrm{B}) = 0,
$$

which is equivalent to the finite horizon optimal policy with $N = 3$. For $\alpha = 0.96$ the policy differs:

$$\mu(\mathrm{T}) = 0, \quad \mu(\mathrm{R}) = 1, \quad \mu(\mathrm{B}) = 1.$$

This policy reflects the fact that when at the bottom of the hill, it is worthwhile to drive and thereby hopefully reach the top of the hill. Note that convergence of VI is slower when $\alpha \to 1$.

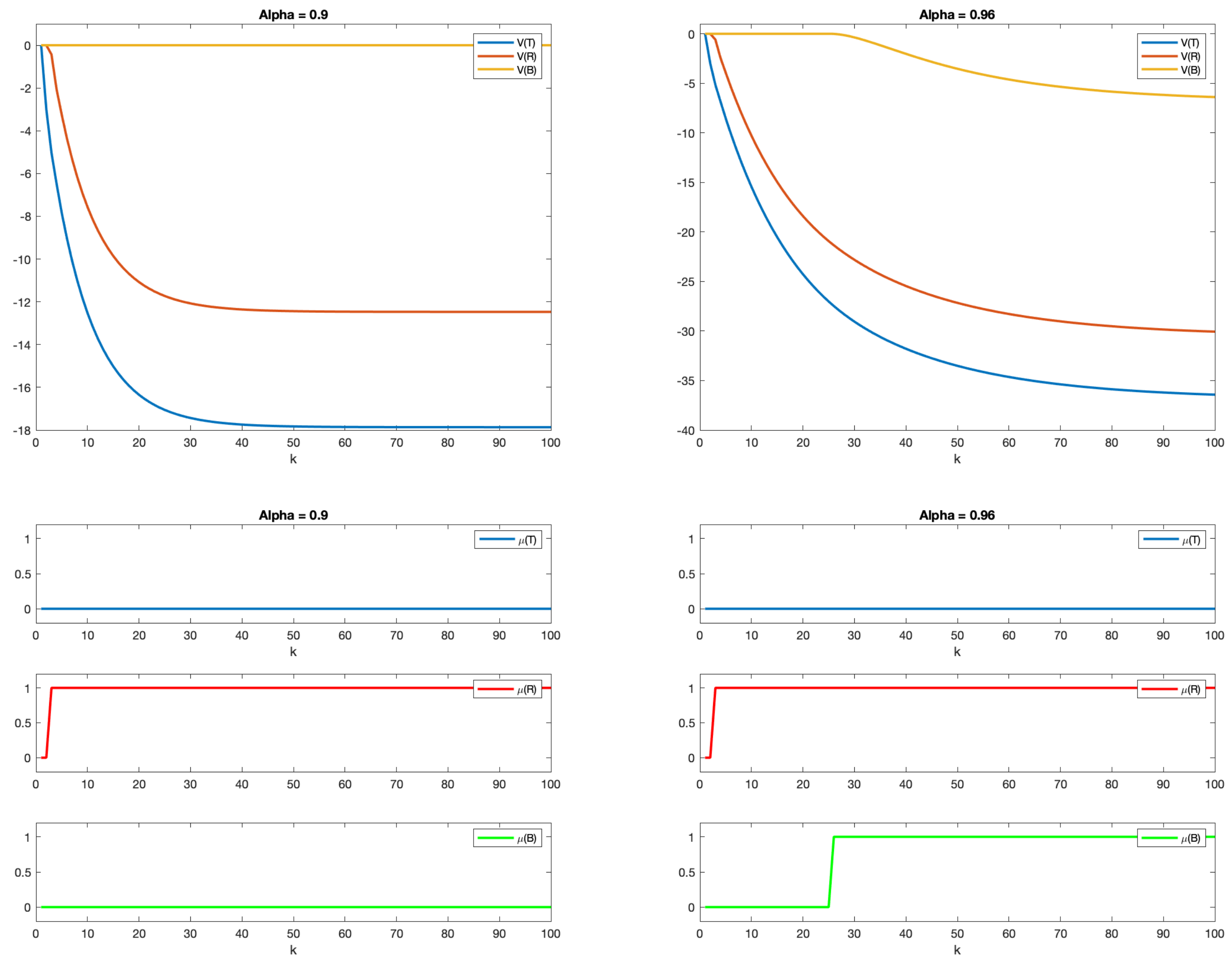

Figure 1: VI for the Rover example: $\alpha = 0.9$ (left), $\alpha = 0.96$ (right)

4 Policy Iteration

We will next illustrate how to evaluate the performance of a given stationary policy $\pi = \{\mu, \mu, \dots\}$. This will be used to formulate the **Policy Iteration** algorithm. From the infinite horizon property (4), it follows directly that the infinite horizon cost of a stationary policy $\pi = \{\mu, \mu, \dots\}$ satisfies its Bellman equation

$$J_\mu(x) = \underset{w}{E}\big\{g(x, \mu(x), w) + \alpha J_\mu\big(f(x, \mu(x), w)\big)\big\}, \quad \forall x \in \mathbb{X}, \tag{16}$$

where we have used the notation J_μ instead of J_π.[3] Similar to the use of operators in (12), we also rewrite the Bellman equation for μ as per:

$$J_\mu(x) = (T_\mu J_\mu)(x), \quad \text{where} \quad (T_\mu J)(x) = \underset{w}{E}\big\{g(x, \mu(x), w) + \alpha J\big(f(x, \mu(x), w)\big)\big\}, \quad \forall x \in \mathbb{X}, \tag{17}$$

The function $T_\mu J$ quantifies the cost of using the policy μ in a one-stage problem with stage cost g and terminal cost αJ.

At this stage it should be of no surprise that the stationary solution $J_\mu(x)$ can typically be calculated recursively using the following **policy evaluation** algorithm, see also [5, pp. 143]:

$$\begin{aligned} J_{\mu,k+1}(x) &= \underset{w}{E}\big\{g(x, \mu(x), w) + \alpha J_{\mu,k}\big(f(x, \mu(x), w)\big)\big\}, \quad \forall x \in \mathbb{X}, \quad k = 0, 1, \dots \\ &= (T_\mu J_{\mu,k})(x) \end{aligned} \tag{18}$$

The observation that a stationary policy π is optimal if and only if for every $x \in \mathbb{X}$, it attains the minimum in Bellman's equation (11), can be used to formulate the **Policy Iteration** (PI) algorithm. Starting with a stationary policy $\pi_0 = \{\mu_0, \mu_0, \dots\}$, PI generates an improving sequence of stationary policies

$$\pi_\ell = \{\mu_\ell, \mu_\ell, \dots\}, \quad \ell = 1, 2, \dots$$

by performing policy evaluation and policy improvement as follows:

Step 1 **Evaluate** the infinite horizon cost of the stationary policy μ_ℓ, This can be done, e.g., using (16) or by performing sufficient iterations of (18). Policy evaluation provides $J_{\mu_\ell}(x)$, $\forall x \in \mathbb{X}$.

Step 2 **Improve** the policy μ_ℓ through the one-step ahead greedy optimisation

$$\mu_{\ell+1}(x) = \underset{u \in \mathbb{U}}{\operatorname{argmin}} \Big\{ \underset{w}{E}\big\{g(x, u, w) + \alpha J_{\mu_\ell}\big(f(x, u, w)\big)\big\}\Big\}, \quad \forall x \in \mathbb{X}, \tag{19}$$

where $J_{\mu_\ell}\big(f(x, u, w)\big)$ is the cost function of the old policy μ_ℓ.
If $\mu_{\ell+1}(x) = \mu_\ell(x)$ for all $x \in \mathbb{X}$, then stop and set $\mu^*(x) \leftarrow \mu_\ell(x)$. Otherwise, set $\ell \leftarrow \ell + 1$ and $\mu_\ell \leftarrow \mu_{\ell+1}$, and go to Step 1.

In contrast to VI, PI finds the optimal policy directly by alternating between policy evaluation and policy improvement. PI can be regarded as an "actor-critic" method, wherein the critic evaluates the current policy (calculates the critique function J_{μ_ℓ}), and the actor takes into account the critique to develop and act out the new stationary policy $\mu_{\ell+1}$.

[3] We shall follow this naming convention when dealing with stationary problems.

Finite Spaces

For finite state spaces $\mathbb{X}$, Policy evaluation through (16) corresponds to solving a system of linear equations, one for each $i \in \mathbb{X}$, cf. (14):

$$J_\mu(i) = g(i, \mu(i)) + \alpha \sum_{j \in \mathbb{X}} p_{ij}(\mu(i)) J_\mu(j), \quad \forall i \in \mathbb{X} \tag{20}$$

and J_μ can be calculated through matrix inversion algorithms.

Alternatively, the recursive policy evaluation algorithm (18) can be used. It takes the form:

$$J_{\mu,k+1}(i) = g(i, \mu(i)) + \alpha \sum_{j \in \mathbb{X}} p_{ij}(\mu(i)) J_{\mu,k}(j), \quad \forall i \in \mathbb{X} \quad k = 0, 1, \ldots. \tag{21}$$

The policy improvement step becomes

$$\mu_{\ell+1}(i) = \operatorname*{argmin}_{u \in \mathbb{U}} \left\{ g(i, u) + \alpha \sum_{j \in \mathbb{X}} p_{ij}(u) J_{\mu_\ell}(j) \right\}, \quad \forall i \in \mathbb{X}$$

and the PI algorithm ends once $\mu_{\ell+1}(i) = \mu_\ell(i)$, for all $i \in \mathbb{X}$.

We note that, since the collection of all stationary policies is finite and every iteration yields an improved policy, the PI algorithm ends in finite time (provided the policy evaluation step does). As an example, for the Rover problem examined before, PI converges within two iterations, see Table 1 and compare to the numerical results in Figure 1. It is worth noting that the optimal policy converges even though the value function has not yet converged.

ℓ	μ_ℓ(T)	μ_ℓ(R)	μ_ℓ(B)	J_{μ_ℓ}(T)	J_{μ_ℓ}(R)	J_{μ_ℓ}(B)
0	0	0	0	-10.2992	0.4073	0.4073
1	0	1	0	-34.3270	-27.6350	0.0449
2	0	1	1	-36.4242	-30.0667	-6.3908
3	0	1	1	-36.4414	-30.0840	-6.4081

Table 1: PI for the Rover problem, with $\alpha = 0.96$ and initial policy $\mu_0(i) = 0$, for all i.

5 Computational Issues and Variations

It can be seen from (9) that each iteration of the VI algorithm involves an optimisation of the decisions $u \in \mathbb{U}$. The obtained decisions are evaluated only over a single iteration step. In contrast, in the PI algorithm, optimal decisions (19) are evaluated over an infinite horizon, using (17) or (18). Thus, policy evaluation in PI requires more computations than policy evaluation in VI. On the other hand, since PI evaluates the infinite horizon behaviour of a policy, for a typical problem instance PI will require less optimisation steps than VI.

From the above discussion, it follows that whether PI is faster than VI or not, will depend on the particular policy evaluation algorithm used. In order to accelerate convergence of PI, a number of variations have been developed. For example, one can limit the number of iterations in the policy evaluation recursions (18); see, e.g., [2, Section 5.2] or [5, Section 6.5].

Whilst PI and VI in their basic form require sweeping the entire state space $\mathbb{X}$ at each iteration, it is possible to formulate algorithm variants, wherein at each time step only some of the states are considered. Policy and cost values at other states are left unchanged. Such methods are termed asynchronous and are especially suited for real-time operation, wherein updates focus on state values that are encountered more often.

6 Average Cost Problems (optional)

As an alternative to discounted costs of the form (3), one can also study average cost problems. Here one seeks to minimise:

$$J_\pi(x) = \lim_{N\to\infty} \frac{1}{N} \underset{w_k}{E}\left\{ \sum_{k=0}^{N-1} g(x_k, \mu_k(x_k), w_k) \right\}.$$

Such cost functions focus on costs occurred in the long run, costs incurred in early stages do not matter. It turns out that most problems of this type, the average cost per stage of a policy is independent of the initial state.

VI and PI algorithms can be formulated for average cost problems as well. This lies outside the scope of the current unit. In case you are interested, please refer to [1, Chapter 4] or also [3, Chapter 5], which presents a more mathematical treatment.

References

[1] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 2. Belmont, MA: Athena Scientific, 2007.

[2] D. P. Bertsekas, *Rollout, Policy Iteration, and Distributed Reinforcement Learning.* Athena Scientific, 2020.

[3] O. Hérnandez-Lerma and J. B. Laserre, *Discrete-Time Markov Control Processes.* New York, N.Y.: Springer-Verlag, 1996.

[4] S. Meyn, *Control Systems and Reinforcement Learning.* Cambridge University Press, 2022.

[5] M. L. Puterman, *Markov Decision Processes.* Hoboken, N.J.: Wiley-Interscience, 1994.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* MIT Press, 2nd ed., 2018.