Chapter 5

Markov decision problems

The present chapter addresses those situations where the decision-maker must make a sequence of decisions in order to complete some target task. We introduce Markov decision problems *as a framework to address sequential decision-making in the face of uncertainty and discuss the concept of optimality in this new framework. We also go over the main algorithmic approaches to solving Markov decision problems.*

5.1 A sequential decision problem

To motivate the problem of sequential decision-making, we revisit the household robot example from Chapter 1.

Example 5.1 We again consider the household robot example in Fig. 1.1, where a robot moves around a household environment assisting human users in their daily activities. Let us suppose that the robot departs from the Hall after receiving a request for assistance in the Kitchen.

In Chapter 4 we modeled this problem using the expected utility theory, framing it as a one-shot decision problem. However, we noted then that the navigation problem was inherently sequential: the agent (robot) selected among several *sequences of steps* leading from the Hall to the Kitchen. For large problems—involving long sequences of steps—the number of possible sequences quickly becomes unmanageable, making this "enumerative" approach impractical.

Alternatively, we can look at each step as an *individual* decision problem. Reaching the goal then requires the agent to solve a *sequence* of decision problems, where the outcome of each influences the next. We model each individual decision in the household robot scenario by adapting the expected utility theory discussed in Chapter 4.

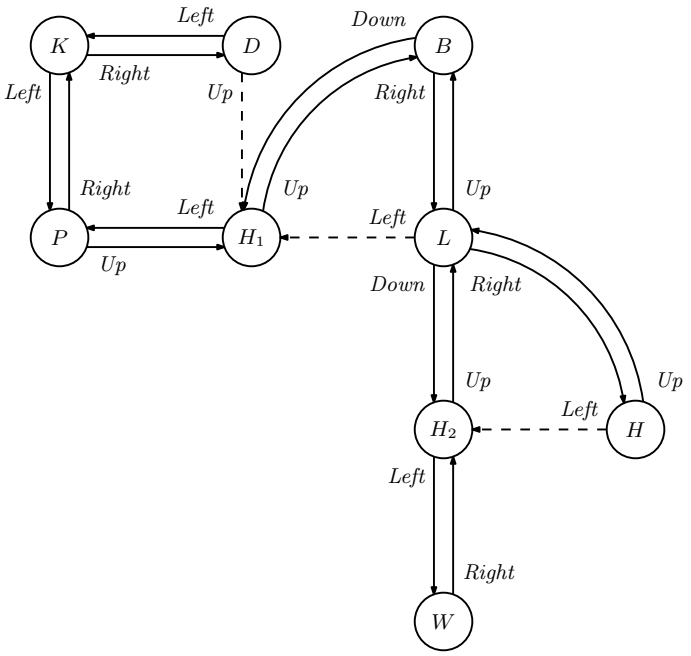


Figure 5.1 Representation of the different actions available to the robot and the corresponding motion between rooms. Dashed lines correspond to motions that may sometimes fail.

Starting with the actions available to the robot, these correspond to the different motions leading out of the robot's current location. For simplicity, let us admit that these actions are the same in all locations, namely *Up* (U), *Down* (D), *Left* (L)and *Right* (R). An action a moves the robot to the adjacent room in the corresponding direction, if there is such a room, or leaves the robot's position unchanged, if no such room exists (see Fig. 5.1 for a depiction of the different actions and corresponding motion).

Additionally, the motion between certain rooms (for example, from the Living Room to the Hallway 1) sometimes fails, leaving the position of the robot unchanged. Such failures occur with a probability 0.4. Finally, we consider one additional action, *Stay* (S), which corresponds to doing nothing. In summary, the set of actions available to the robot at each step is

$$\mathcal{A} = \{U, D, L, R, S\}\,.$$

The outcomes associated with each action $a \in \mathcal{A}$ correspond to the possible locations to which the robot may move as a result of executing a. In other words, the set of possible outcomes is

$$\mathcal{X} = \{B, D, H, H_1, H_2, K, L, P, W\}$$

although, depending on the location of the robot, only some of the outcomes are possible. Let x be the r.v. corresponding to the location of the robot and a the r.v. corresponding to the action of the robot at some time step t, and let x' be the r.v. corresponding to the location of the robot as a result of a. We can adapt the notation in Chapter 4 and let

$$P(y \mid x, a) \stackrel{\text{def}}{=} \mathbb{P}\left[\mathrm{x}' = y \mid \mathrm{x} = x, \mathrm{a} = a\right].$$

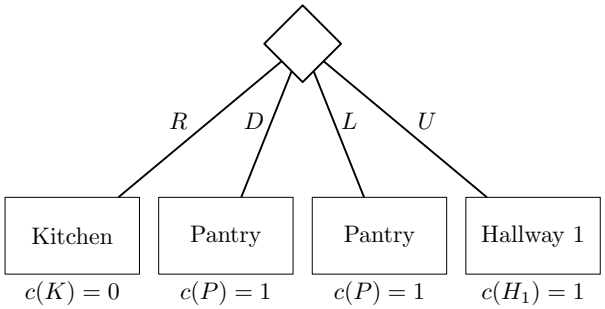


Figure 5.2 Decision tree for the robot when located at the Living Room.

In other words, $P(y \mid x, a)$ denotes the probability of outcome y (i.e., the robot moving to location y) as a result of executing action a when the robot is in location x. For example, from our previous description we have that

$$\begin{aligned} P(\mathrm{H1} \mid \mathrm{LR}, R) &= 0.6, \\ P(\mathrm{LR} \mid \mathrm{LR}, R) &= 0.4, \\ P(\mathrm{H2} \mid \mathrm{H2}, D) &= 1.0, \\ P(\mathrm{K} \mid \mathrm{D}, R) &= 1.0. \end{aligned}$$

Finally, the decision problem is completely specified as soon as we define the cost translating our preferences regarding the behavior of the robot. In this particular scenario, we want the robot to select the actions leading to the Kitchen. Therefore—and adhering to the discussion in in Section 4.4—one possible cost function is to set $c(x) = 1 - \mathbb{I}\left[x = K\right]$, where any outcome (other than the Kitchen) incurs a cost of 1.

With this model in place, we can now apply the expected utility theory to the individual decisions faced by the robot at each step. Let us consider, for example, that the robot is the Pantry at some time step t. The decision tree for the robot is depicted in Fig. 5.2. Since there is no uncertainty, the decision is clear: the robot should select the action R, since it is the one minimizing the incurred cost.

Let us now consider the case where the robot, at time step t, is in the Living Room. The corresponding decision tree is depicted in Fig. 5.3. Unlike the previous situation, none of the actions leads directly to the desired outcome.

Therefore, all yield the same expected cost, and it is not possible, from this information alone, to determine which is the best action to take at this state.

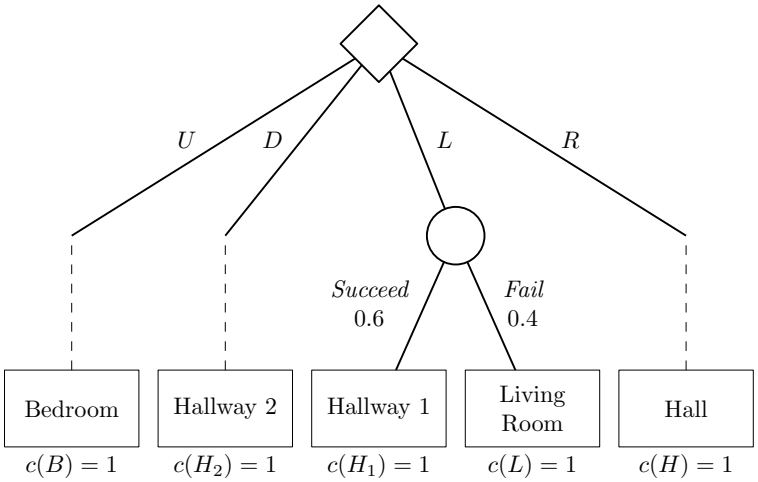


Figure 5.3 Decision tree for the robot when located at the Living Room.

Example 5.1 prompts several important observations. First, the decision of the robot depends on its location. For example, when the robot is in the Pantry, the best action is clearly *Up*. If the robot is in the Living Room, it is no longer clear whether *Up* is the best action to take. And, by inspection, we can easily venture that if the robot is in Hallway 1, *Up* is not the best action to take. In terms of the formalism introduced in Chapter 4, the *policy* of the robot at each step t depends on its position at that time step.[1] As will soon become apparent, this is not specific of Example 5.1—sequential decision problems will generally require policies that depend on the situation of the agent.

A second observation is that, while the cost function selected does convey the necessary information about our preferences over the possible outcomes of the robot's actions, it is not sufficiently informative for the robot to select its actions. In a sense, the cost function provides immediate information about the cost of its actions, but not about their long-term cost. To emphasize that fact, we henceforth refer to the cost function c as the *immediate cost*. A significant effort in solving sequential decision problems goes, precisely, to the computation of the *long-term costs* incurred by an agent's actions.

Finally, note that if we fix the actions selected by the robot, its position evolves as a *Markov chain*: the position of the robot at any time step $t + 1$ depends only on its position at time step t. In this chapter, we focus on sequential decision problems that, much like the one in Example 5.1, possess the Markov property. Such problems are known as *Markov decision problems*.

[1] Recall from Chapter 4 that a *policy* is a distribution π over the set of actions available to the agent.

5.2 Markov decision problems

Consider an agent engaging in a sequence of decision problems. We index each decision by a time index $t \in \mathcal{T}$, where $\mathcal{T} \subset \mathbb{N}$, and refer to each t as a *decision epoch*. Let $\{\mathrm{x}_t, t \in \mathcal{T}\}$ denote a stochastic process that evolves along the agent's decisions. In particular, at each time step $t \in \mathcal{T}$, the agent has available a set of actions that depends on x_t, while each x_t depends (probabilistically) on the agent's actions up to time-step $t-1$.

Each r.v. x_t takes values in some finite set $\mathcal{X}$ and is henceforth referred as the *state* of the decision process at time step t. The set $\mathcal{X}$ is the *state space*. Similarly, we represent the action of the agent at time step t as a r.v. a_t that takes values in some finite set $\mathcal{A}(\mathrm{x}_t)$, and refer to the set

$$\mathcal{A} \stackrel{\text{def}}{=} \bigcup_{x \in \mathcal{X}} \mathcal{A}(x)$$

as the *action space*. We write h_t to denote the *history* of the process up to time step t, defined as the r.v.

$$\mathrm{h}_t = \{\mathrm{x}_0, \mathrm{a}_0, \ldots, \mathrm{x}_{t-1}, \mathrm{a}_{t-1}, \mathrm{x}_t\},$$

and refer to t as the *length* of h_t, denoted as $|h|$. We write $\mathcal{H}_t$ to denote the set of all t-length histories, and let

$$\mathcal{H} \stackrel{\text{def}}{=} \bigcup_{t \in \mathcal{T}} \mathcal{H}_t.$$

Finally, let $c : \mathcal{H} \times \mathcal{A} \to \mathbb{R}$ denote an *immediate cost function*. At each time step $t \in \mathcal{T}$, and upon executing an action $a \in \mathcal{A}(\mathrm{x}_t)$, the agent incurs a cost c_t given by

$$\mathrm{c}_t = c(\mathrm{h}_t, a).$$

We refer to the process $\{(\mathrm{x}_t, \mathrm{a}_t, \mathrm{c}_t), t \in \mathcal{T}\}$ thus defined as a *sequential decision process*. Note, in particular, that the decision processes discussed in Chapter 4 can be seen as a particular case of sequential decision processes.

In the remainder of the chapter, and except when stated otherwise, we admit that $\mathcal{T} = \mathbb{N}$ and that the actions available to the agent at each time step t are independent of x_t. The latter assumption in particular greatly simplifies the presentation without affecting the representational power of the model, since it is always possible to add dummy actions to any of the sets $\mathcal{A}(x)$.

$\diamond$

In the remainder of the chapter, we focus on decision processes for which the *Markov property* holds.

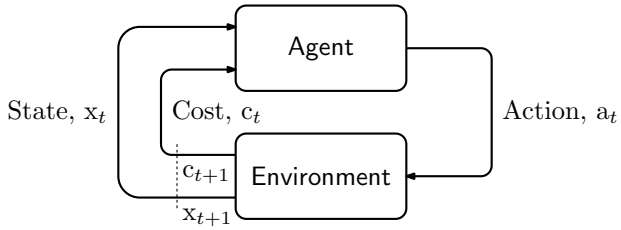


Figure 5.4 Diagram representing the interaction between the agent and the environment in a Markov decision process. Diagram adapted from the book of Sutton and Barto (1998).

Markov property

A decision processes $\{(\mathrm{x}_t, \mathrm{a}_t, \mathrm{c}_t), t \in \mathbb{N}\}$ verifies the *Markov property* if, for any $h \in \mathcal{H}_t$ such that

$$h = \{x_0, a_0, \ldots, x_{t-1}, a_{t-1}, x\},$$

it holds that

$$\mathbb{P}\left[\mathrm{x}_{t+1} = y \mid \mathrm{h}_t = h\right] = \mathbb{P}\left[\mathrm{x}_{t+1} = y \mid \mathrm{x}_t = x, \mathrm{a}_t = a\right]$$

and

$$\mathrm{c}_t \stackrel{\text{def}}{=} c(\mathrm{h}_t, \mathrm{a}_t) = c(\mathrm{x}_t, \mathrm{a}_t),$$

for all $y \in \mathcal{X}$ and $a \in \mathcal{A}$. A decision process verifying the Markov property is known as a *Markov decision processes*.

In a Markov decision process, the state process $\{\mathrm{x}_t, t \in \mathbb{N}\}$ is a *controlled Markov chain* with state space $\mathcal{X}$ and in which the transition probabilities depend on the action of the agent. The interaction between agent and environment is depicted in the diagram of Fig. 5.4. For each $a \in \mathcal{A}$, we define the *transition probability matrix* $\boldsymbol{P}_a$ as

$$[\boldsymbol{P}_a]_{x,y} \stackrel{\text{def}}{=} \mathbb{P}\left[\mathrm{x}_{t+1} = y \mid \mathrm{x}_t = x, \mathrm{a}_t = a\right], \tag{5.1}$$

with $x, y \in \mathcal{X}$. We denote the transition probability in (5.1) interchangeably as $\boldsymbol{P}_a(y \mid x)$—when we want to emphasize its nature as a transition probability associated with the action a—or as $\boldsymbol{P}(y \mid x, a)$—when we want to a emphasize its nature as a transition probability of a controlled Markov chain.

A Markov decision process is fully specified by

- Its state space, $\mathcal{X}$;
- Its action space, $\mathcal{A}$;

- Its transition probabilities, $\boldsymbol{P}_a, a \in \mathcal{A}$;
- The immediate cost function, c;
- The initial state, x_0.

We henceforth refer to a Markov decision process as a tuple $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, x_0)$ or, when the initial state is implicit or immaterial for the discussion, using the more compact representation $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c)$.

5.2.1 Policies

We now extend the notion of policy, first introduced in Chapter 4, to sequential decision processes.

A *policy* is a mapping $\pi : \mathcal{H} \to \Delta(\mathcal{A})$ where, as before, $\Delta(\mathcal{A})$ is the family of all probability distributions over $\mathcal{A}$. In other words, a policy π maps each history $h \in \mathcal{H}$ to a distribution $\pi(h)$ over $\mathcal{A}$. Note that, in the one-shot decision problems discussed in Chapter 4, the two definitions of policy coincide.

Given a policy π, we denote by $\pi(a \mid h)$ the probability of action $a \in \mathcal{A}$ given history h, according to policy π. We say that an agent follows policy a π if

$$\mathbb{P}\left[\mathrm{a}_t = a \mid \mathrm{h}_t = h\right] = \pi(a \mid h).$$

Policies can be categorized according to different criteria.

Policy categorization

A policy π is

- *Deterministic* if, for each history $h \in \mathcal{H}$, there is an action $a \in \mathcal{A}$ such that $\pi(a \mid h) = 1$. If π is deterministic, we compactly write $\pi(h)$ to denote the single action $a \in \mathcal{A}$ such that $\pi(a \mid h) = 1$.
- *Stochastic* or *randomized*, if it is not deterministic.
- *Markov*, if the probability $\pi(a \mid h)$ depends only on $|h|$ and on the last state in h. If π is Markov, we write $\pi_t(a \mid x)$ to compactly denote the probability $\pi(a \mid h)$ when $|h| = t$ and x is the last state visited in h.
- *Stationary* if it is Markov and the probability $\pi(a \mid h)$ does not depend on $|h|$ but only on the last state in h. If π is stationary, we write $\pi(a \mid x)$ to denote the probability $\pi(a \mid h)$ when x is the last state visited in h.
- *History-dependent*, if it is not Markov.

We write

- Π^{HR} to denote the set of history-dependent and randomized policies;
- Π^{MR} to denote the set of all Markov and randomized policies;
- Π^{SR} to denote the set of all stationary and randomized policies;
- Π^{HD} to denote the set of history-dependent and deterministic policies;
- Π^{MD} to denote the set of all Markov and deterministic policies;
- Π^{SD} to denote the set of all stationary and deterministic policies,

with the obvious inclusions

$$\Pi^{\mathrm{S}\cdot} \subset \Pi^{\mathrm{M}\cdot} \subset \Pi^{\mathrm{H}\cdot},$$

where $\cdot$ can be any of D or R, and

$$\Pi^{\cdot\mathrm{D}} \subset \Pi^{\cdot\mathrm{R}},$$

where $\cdot$ can be any of S, M or R.

Given a Markov decision process $\{(\mathrm{x}_t, \mathrm{a}_t, \mathrm{c}_t), t \in \mathbb{N}\}$, if the agent follows a fixed Markov policy π, the resulting process $\{(\mathrm{x}_t, \mathrm{c}_t), t \in \mathbb{N}\}$ is called a *Markov cost process*.[2] The associated state process, $\{\mathrm{x}_t, t \in \mathbb{N}\}$, is a standard Markov chain with transition probabilities

$$\mathbb{P}\left[\mathrm{x}_{t+1} = y \mid \mathrm{x}_t = x\right] = \sum_{a \in \mathcal{A}} \pi_t(a \mid x) \boldsymbol{P}_a(y \mid x),$$

and the associated cost process, $\{\mathrm{c}_t, t \in \mathbb{N}\}$ is governed by the probabilities

$$\mathbb{P}\left[\mathrm{c}_t = c \mid \mathrm{x}_t = x\right] = \sum_{a \in \mathcal{A}} \pi_t(a \mid x) \mathbb{I}\left[c(x, a) = c\right].$$

Computationally, it is often more convenient to work with the *expected* cost at each step t, given by

$$\hat{c}_t = \sum_{a \in \mathcal{A}} \pi_t(a \mid \mathrm{x}_t) c(\mathrm{x}_t, a).$$

If, moreover, π is also stationary, then the Markov chain $\{\mathrm{x}_t, t \in \mathbb{N}\}$ is time-homogeneous, with transition probability matrix $\boldsymbol{P}_\pi$ given by

$$[\boldsymbol{P}_\pi]_{x,y} \stackrel{\mathrm{def}}{=} \mathbb{P}\left[\mathrm{x}_{t+1} = y \mid \mathrm{x}_t = x\right] = \sum_{a \in \mathcal{A}} \pi(a \mid x) \boldsymbol{P}_a(y \mid x),$$

and we can define the function $c_\pi : \mathcal{X} \to [0, 1]$ as

$$c_\pi(x) \stackrel{\mathrm{def}}{=} \sum_{a \in \mathcal{A}} \pi(a \mid x) c(x, a),$$

where the expected cost $\hat{c}_t$ is the image of x_t under c_π for all $t \in \mathbb{N}$.

[2] In many MDP texts, the cost function is replaced by a *reward* function. In that case, the process induced by a fixed policy is known as a *Markov reward processes*.

5.2.2 Discounted cost-to-go

As seen in Section 5.1, the cost function c provides the agent with instantaneous feedback regarding the outcome of its actions. Such instantaneous feedback, however, is often insufficient to determine the best action to take at each step t, since it conveys no long-term information regarding the agent's task.

Several long term criteria can be used to guide the agent's actions. The one most commonly found in the literature is the *total expected discounted cost-to-go* criterion. We adopt such criterion throughout the book, and refer to Section 5.6 for a brief discussion of alternative optimality criteria.

The *expected discounted cost-to-go* (EDC) associated with a Markov decision process $\{(\mathrm{x}_t, \mathrm{a}_t, \mathrm{c}_t), t \in \mathbb{N}\}$ is the quantity

$$DC \stackrel{\text{def}}{=} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathrm{c}_t\right], \tag{5.2}$$

where $\gamma \in [0, 1)$. We can interpret the expression in (5.2) as the (expected) total cost incurred by the agent, but where costs arising in the future are less important than those occurring immediately. The scalar γ is both mathematically convenient (since $\mathrm{c}_t \in [0, 1]$, the quantity in (5.2) is well-defined) and has an economic interpretation: it plays the role of an "inflation rate".

The EDC provides a criterion to compare policies, and is thus called an *optimality criterion*. The agent faces the decision problem of selecting a policy that optimizes the EDC. In general, the decision problem resulting from augmenting a Markov decision process with an optimality criterion—such as the EDC in (5.2)—is called a *Markov decision problem* (MDP). We compactly represent an MDP with the EDC as optimality criterion as a tuple $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$.

Given an MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$, we define the cost-to-go associated with a policy π and a state $x \in \mathcal{X}$ as the quantity

$$J^\pi(x) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t \mathrm{c}_t \mid \mathrm{x}_0 = x\right], \tag{5.3}$$

where the expectation is taken with respect to the distribution over trajectories $\{\mathrm{c}_t, t \in \mathbb{N}\}$ induced by policy π (see Exercise 5.1). The function $J^\pi : \mathcal{X} \to \mathbb{R}$ defined in (5.3) is known as the *cost-to-go function* associated with policy π. A policy π^* is *optimal* if, for all $x \in \mathcal{X}$,

$$J^{\pi^*}(x) \leq J^\pi(x),$$

for any policy $\pi \in \Pi^{\text{HR}}$. The cost-to-go function associated with an optimal policy π^* is denoted as J^* and verifies, for all $x \in \mathcal{X}$,

$$J^*(x) = \inf_{\pi \in \Pi^{\text{HR}}} J^\pi(x) = \inf_{\pi \in \Pi^{\text{HR}}} \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t \mathrm{c}_t \mid \mathrm{x}_0 = x\right]. \tag{5.4}$$

We conclude this section by going over several MDP examples that illustrate how to construct the model for this class of problems. The following section the

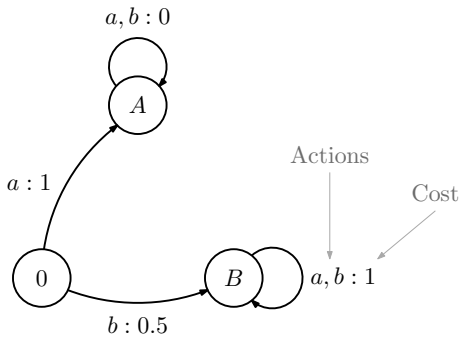


Figure 5.5 Transition diagram for a simple 3-state MDP. The representation extends that adopted in Chapter 2 for Markov chains. In particular, states correspond to nodes in the diagram. Edges correspond to transitions between states. Each edge is labeled with two pieces of information: a label of the form $a : c$, where a is the action corresponding to the transition, and c is the associated cost; and a numeric label with the probability associated to the transition. Transitions with probability 0 are not drawn; transitions with probability 1 do not include the probability label.

proceeds by establishing the key result in this chapter, providing a definitive and satisfying answer to the following question: does an optimal policy exist for a given an MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$?

5.2.3 Examples

We now go over some MDP examples.

3-state MDP

We start with a very simple 3-state MDP, depicted in Fig. 5.5 In this process, the agent has essentially one decision to make, concerning the action in state 0. In states A and B both actions yield the same transitions and cost, so the agent is indifferent between the two actions.

We can model this decision problem as an MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$, where

- The state space is $\mathcal{X} = \{0, A, B\}$;
- The action space is $\mathcal{A} = \{a, b\}$;
- The transition probabilities are summarized in the transition matrices

$$\boldsymbol{P}_a = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{P}_b = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix};$$

- The cost function can be succinctly represented as a matrix

$$\boldsymbol{C} = \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 0 \\ 1 & 1 \end{bmatrix},$$

  where the $C_{x,a}$ corresponds to the cost $c(x, a)$.

Let us now consider two stationary policies, π and π', which we write in matrix form as

$$\boldsymbol{\pi} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{\pi}' = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

As with $\boldsymbol{C}$, the component $\pi_{x,a}$ of $\boldsymbol{\pi}$ corresponds to the probability $\pi(a \mid x)$. Therefore, policy π always selects action a, while policy π' always selects action b.

Let us first consider the case where $\gamma = 0.99$. If the agent starts in state A or B, the total discounted cost incurred is independent of the policy. Therefore, we have that

$$J^{\pi}(A) = J^{\pi'}(A) = \sum_{t=0}^{\infty} \gamma^t \times 0 = 0,$$

and

$$J^{\pi}(B) = J^{\pi'}(B) = \sum_{t=0}^{\infty} \gamma^t \times 1 = \frac{1}{1-\gamma} = 100.$$

The cost-to-go at state 0, on the other hand, does depend on the policy. Since the MDP in Fig. 5.5 is deterministic, we immediately get

$$J^{\pi}(0) = 1 + \sum_{t=1}^{\infty} \gamma^t \times 0 = 1$$

and

$$J^{\pi'}(0) = 0.5 + \sum_{t=1}^{\infty} \gamma^t \times 1 = 0.5 + \frac{\gamma}{1-\gamma} = 99.5.$$

We can represent J^{π} and $J^{\pi'}$ as matrices, yielding

$$\boldsymbol{J}^{\pi} = \begin{bmatrix} 1 \\ 0 \\ 100 \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{J}^{\pi'} = \begin{bmatrix} 99.5 \\ 0 \\ 100 \end{bmatrix}.$$

It follows that policy π is better than policy π', since $J^{\pi}(x) \leq J^{\pi'}(x)$ for all $x \in \mathcal{X}$. In fact, it is not difficult to conclude that the policy π is optimal for this MDP.

Let us now consider the case of a generic discount factor γ. The essence of the reasoning above holds. The only distinction is in the conclusion: policy π is better than policy π' only if

$$J^{\pi}(0) < J^{\pi'}(0)$$

or, equivalently, if

$$1 < 0.5 + \frac{\gamma}{1-\gamma}.$$

Therefore, for small values of γ ($\gamma < 0.33$), future costs have little impact in the total cost. Therefore, the agent is better off trying to minimize the immediate cost, selecting action b in state 0.

Hiring a computer engineer

This second example illustrates an approach to model "memory" in a Markov setup.

A manager in an IT company is tasked with recruiting a new computer engineer. After an initial selection process, a total of N candidates were shortlisted and called for an interview. The candidates are interviewed sequentially, and the manager must decide, by the end of each interview, whether or not to offer the job to the candidate being interviewed (the candidate is lost otherwise). If no candidate has been hired by the time that the last candidate is interviewed, the employer will always choose to hire that candidate.

The manager has access to an autonomous decision support system that takes as input the number of the candidate and an assessment (inserted by the manager) of whether the current candidate is the best interviewed so far. The decision support system must recommend the manager whether or not to hire the current candidate.

We model the reasoning process of the autonomous decision support system (the agent) as an MDP in which the goal is to recommend hiring the best candidate. Decision epochs occur immediately after each interview, at which time the system must output a recommendation.

One first approach to modeling the decision problem is to consider only three states: "Best so far" (B), "Not best so far" ($\bar{B}$) and "Hired" (H), where the latter state corresponds to the situation in which an engineer has already been hired. The actions available are "Hire current candidate" (H) or not ($\bar{H}$). As for the transition probabilities, if $\mathrm{a}_t = H$ at some time step T, then $\mathrm{x}_t = H$ for all $t > T$. For $t < T$, the probability that the next candidate (candidate $t+1$) is the best among the first $t+1$ candidates is $1/(t+1)$, and does not depend on x_t.

Even before considering the immediate cost, it becomes apparent that the transition probabilities (and hence the decision of the agent) depend on how many candidates were interviewed so far. Therefore, the state of the decision process should include information regarding the number of the candidate—in other words, it should "remember" how many candidates were already interviewed at each time step.

Let us then consider an extended state space, where each state includes both the number of the candidate and whether that candidate is the best candidate so far. The resulting MDP is represented in the transition diagram of Fig. 5.6, where the transition probabilities correspond to the ones discussed above.[3] Note that if

[3]Recall that each transition includes two labels: a label of the form $a : c$, where a is the action corresponding to that transition and c is the corresponding cost, and a label p with the probability associated with the transition.

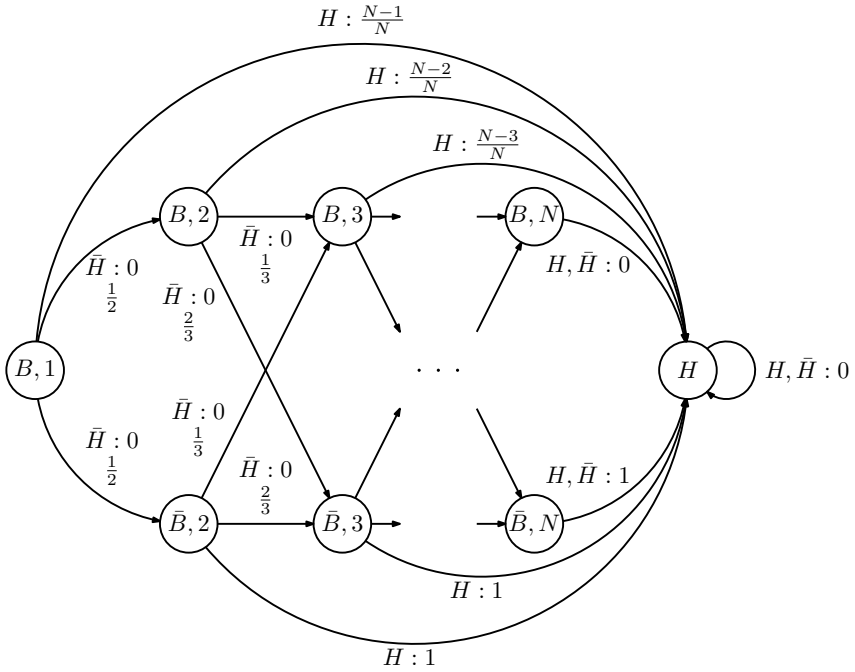


Figure 5.6 Transition diagram for the hiring example. See text for details. Each transition is labeled with the corresponding action, cost and transition probabilities. Transitions with probability 1 do not include the transition probability.

the interview process reaches the Nth candidate, the only available action is H. However, to simplify the modeling process, we still include both actions (H and $\bar{H}$) and make them equivalent. Similarly, after hiring a candidate, no more actions are available. However, we still include both actions in state H but having no effect.

As for the costs, let us suppose that a candidate is hired at time step t. If the current candidate is not the best so far, then certainly it is not the best candidate. Hiring such candidate has, in this case, the maximum cost (1). If it is the best candidate so far, the probability that it is the best overall is given by t/N. Therefore, we associate with the action of hiring the tth candidate knowing that it is the best so far a cost of $1 - t/N$. For the particular case where the Nth candidate interviewed is the best so far, the agent can be certain that it is the best candidate overall, so hiring has a cost of 0.

The resulting MDP is thus a tuple $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$, where

- The state space is $\mathcal{X} = \big\{(B,1), (B,2), (\bar{B},2), \ldots, (B,N), (\bar{B},N), H\big\}$;
- The action space is $\mathcal{A} = \big\{H, \bar{H}\big\}$;

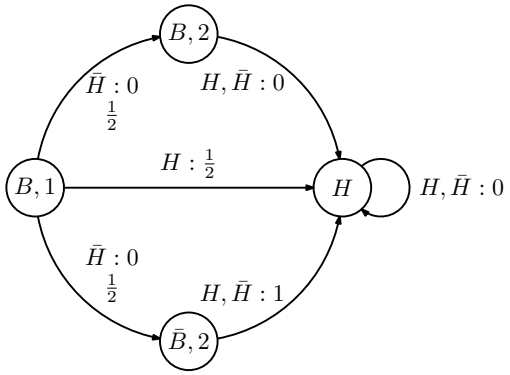


Figure 5.7 Transition diagram for the hiring example when $N = 2$.

- The transition probabilities for the action H are simply

$$\boldsymbol{P}_H(y \mid x) = \begin{cases} 1 & \text{if } y = H \\ 0 & \text{otherwise,} \end{cases}$$

  for any $x \in \mathcal{X}$. As for the action $\bar{H}$,

$$\boldsymbol{P}_{\bar{H}}(y \mid x) = \begin{cases} \frac{1}{t+1} & \text{if } x = (\cdot, t),\ y = (B, t+1) \text{ and } t < N, \\ \frac{t}{t+1} & \text{if } x = (\cdot, t),\ y = (\bar{B}, t+1) \text{ and } t < N, \\ 0 & \text{if } x = (\cdot, t),\ y \not= (\cdot, t+1) \text{ and } t < N, \\ 1 & \text{otherwise,} \end{cases}$$

  where $\cdot$ can be any of B or $\bar{B}$.

- Finally, the cost can be expressed as

$$c(x, \bar{H}) = \begin{cases} 1 & \text{if } x = (\bar{B}, N), \\ 0 & \text{otherwise} \end{cases}$$

  and

$$c(x, H) = \begin{cases} \frac{N-t}{N} & \text{if } x = (B, t), \text{ for all } t, \\ 0 & \text{otherwise.} \end{cases}$$

Let us again compare two stationary policies, π and π', in the simple situation where $N = 2$. In this case, the transition diagram in Fig. 5.6 becomes the simpler diagram in Fig. 5.7. We analyze the policies

$$\boldsymbol{\pi} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{\pi}' = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

Policy π always selects action H, while policy π' always selects action $\bar{H}$. Note, however, that the action in states $(B, 2)$, $(\bar{B}, 2)$ and H is irrelevant, since all actions are equivalent.

Consider the case where $\gamma = 0.99$. We have that

$$J^{\pi}(H) = J^{\pi'}(H) = 0.$$

Similarly,

$$J^{\pi}(B, 2) = J^{\pi'}(B, 2) = 0$$

and

$$J^{\pi}(\bar{B}, 2) = J^{\pi'}(\bar{B}, 2) = 1.$$

Finally, we have that

$$J^{\pi}(B, 1) = 0.5 + \frac{\gamma}{1 - \gamma} \times 0 = 0.5.$$

To determine $J^{\pi'}(B, 1)$, we must consider the result of selecting $\bar{H}$ in state $(B, 1)$. With a probability 0.5, the agent goes to state $(B, 2)$, after which it transitions to state H and remains there. Conversely, with probability 0.5, the agent goes to state $(\bar{B}, 2)$, from which it incurs a cost of 1 and transitions to H, remaining there afterwards. Summarizing,

$$J^{\pi'}(B, 1) = 0.5 \times \underbrace{(0 + \gamma \times 0 + \frac{\gamma^2}{1 - \gamma} \times 0)}_{\text{Transitions to } (B, 2)} + 0.5 \times \underbrace{(0 + \gamma \times 1 + \frac{\gamma^2}{1 - \gamma} \times 0)}_{\text{Transitions to } (\bar{B}, 2)} =$$

$$= 0 + 0.495 = 0.495.$$

Since $J^{\pi}(x) \geq J^{\pi'}(x)$ for all $x \in \mathcal{X}$, policy π' is preferable to policy π. As in the previous example, it is not hard to conclude that, in fact, π' is the optimal policy for the present problem.

The result above indicates that the manager should never hire the first candidate. In other words, the interview process is unnecessary: it is always better to hire the second candidate. This result can easily be interpreted if we consider that—in both policies—the probability of hiring the best candidate is 0.5. Therefore, at first sight, we would expect both policies have the same cost. However, since policy π' hires the candidate one step later, the discount γ implies that the cost incurred by π' is slightly smaller.

The above argument also implies that, unlike the previous example, the optimal policy is independent of the value of γ, as long as $\gamma < 1$. In fact, if $\gamma < 1$, it is always better to hire the second candidate (again, independently of the interview process). In subsequent sections, as we develop solution methods for MDPs, we revisit this example to observe more complex interactions occurring when $N > 2$.

$\diamond$

The next examples illustrate the process of modeling more complex decision problems using the MDP framework. We postpone to Section 5.4 the analysis of the corresponding optimal policies.

Epidemic control

A population of N individuals has been exposed to an infectious disease. At every time step t, a new individual is infected with a probability that is proportional both to the number of infected individuals and the number of healthy individuals. On the other hand, the probability of an infected individual recovering from the infection is proportional to the number of healthy individuals in the population.

We want to design an autonomous (decision support) agent to fight the infection and prevent its spreading. The possible interventions include:

- Quarantine measures, i.e., holding in isolation a percentage of the total population. Quarantine measures reduce the appearance of new infections.
- Deploy medical units to treat a percentage of the infected population; treatment measures affect both the rate of contagion and the rate at which healing takes place.

Infected individuals have a social cost (they are not suitable for work and pose a greater risk of an epidemic). On the other hand, both intervention options (quarantine and treatment) are expensive. The agent must determine the best way to act so as to minimize the economic and health impact of the epidemic.

We model this problem as an MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$, where

- The state at time step t, x_t, is the number of infected individuals, taking values in the set $\mathcal{X} = \{1, \ldots, N\}$;
- The action space encompasses both intervention possibilities. The agent can decide to quarantine 0, 10, 50 or 100 percent of the population. Similarly, it may choose to deploy treatment stations covering 0, 10, 50 or 100 percent of the affected area. The action space is, thus, $\mathcal{A} = \{0.0, 0.1, 0.5, 1.0\} \times \{0.0, 0.1, 0.5, 1.0\}$. We represent each action in $\mathcal{A}$ as a pair $\boldsymbol{a} = (a_Q, a_T)$, where a_Q and a_T denote the percentage of quarantined population and area covered by the treatment stations, respectively.
- The transition probabilities follow the model just described. In particular, for $x < N$ we get

  $$\boldsymbol{P}(x+1 \mid x, \boldsymbol{a})) = K\big(\lambda_I(x, a_Q) + \lambda_C(x, a_T)\big),$$

  where $\lambda_I(x, a_Q)$ is the infection rate when there are x individuals infected and a level a_Q of quarantine is in effect; $\lambda_C(x, a_Q)$ is the contagion rate when there are x individuals infected and a level a_T of treatment is in effect; K is a normalization constant. Similarly,

  $$\boldsymbol{P}(x-1 \mid x, \boldsymbol{a}) = K \cdot \mu(x, a_T),$$

  where μ is the healing rate. For the purpose of this example, we use

  $$\lambda_I(x, a_Q) = (N - x)\lambda_0 e^{-3a_Q},$$

i.e., infections increase with the number of healthy individuals, $(N - x)$;

$$\lambda_C(x, a_T) = (N - x)x\lambda_0 e^{-3a_T},$$

i.e., contagion increases with both the number of infected and healthy individuals; and

$$\mu(x, a_T) = x\mu_0 a_T(2 - a_T),$$

i.e., healing increases with the number of healthy individuals.

- The immediate cost is given by

$$c(x, \boldsymbol{a}) = c_Q\, a_Q^2 + c_T\, a_T\, x + c_I\, x^3,$$

where c_Q, c_T and c_I are constants that weight the cost of quarantine, treatment and infected people.

Fault detection

Consider the problem of developing an automated fault detection mechanism for a complex system. The agent (in this case, the fault detection mechanism) must determine which module, in the system, is responsible for its failure as a whole. The system includes a total of N modules; each module n is known to fail with a probability $p_n, n = 1, \ldots, N$, and takes t_n time to inspect. For simplicity, we assume that no two modules are at fault simultaneously.

We model the detection process as an MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$. A new decision epoch occurs every time a module is inspected; the agent must decide which module to inspect next. The relevant information for the decision of the agent is the state of each module in the system. The state of the MDP is the joint state of all modules in the system. Each module can be "Faulty" (F), "Not faulty" $(\bar{F})$ or "Unknown" (U), i.e., the state of the untested modules. The state of the system at time step t is, therefore, a random vector

$$\mathbf{x}_t = [\mathrm{x}_{1,t}, \ldots, \mathrm{x}_{n,t}],$$

where $\mathrm{x}_{n,t}$ is the state of module n at time step t. At each decision epoch, the agent can select one of the N modules to inspect. Therefore, the agent has a total of N actions, $\mathcal{A} = \{1, \ldots, N\}$, where action n corresponds to inspecting module n.

Let

$$\begin{aligned} P_n(y \mid x, m) &\stackrel{\text{def}}{=} \mathbb{P}\left[\mathrm{x}_{n,t} = y \mid \mathrm{x}_{n,t} = x, \mathrm{a}_t = m\right] \\ &= \begin{cases} 1 & \text{if } x = y \text{ and } (x \neq U \text{ or } m \neq n); \\ p_n & \text{if } x = U \text{ and } m = n \text{ and } y = F; \\ 1 - p_n & \text{if } x = U \text{ and } m = n \text{ and } y = \bar{F}; \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

The transition probabilities can be written as

$$\boldsymbol{P}(\boldsymbol{y} \mid \boldsymbol{x}, m) = \prod_{n=1}^{N} P_n(y_n \mid x_n, m).$$

The transition probabilities can be written as

$$\boldsymbol{P}(\boldsymbol{y} \mid \boldsymbol{x}, m) = \prod_{n=1}^{N} P_n(y_n \mid x_n, m).$$

Finally, we assume that as soon as a faulty module is detected, the agent incurs no additional cost. The cost of each inspection action n is directly given by the time it takes the inspection of module n. In other words,

$$c(\boldsymbol{x}, n) = \frac{t_n}{\max_m t_m}$$

for all $\boldsymbol{x}$ such $x_n \neq F, n = 1, \ldots, N$.

Bus scheduling

A bus serves two locations, A and B. Location A is the bus station, where the bust is usually stationed, while B is a regular stop. We want to design an automated dispatcher to decide when to dispatch the bus from A. When dispatched, the bus travels from A to B carrying any passengers waiting at A at the time of departure. It stops in B only for as much time as necessary to drop the passengers it carries and pick up new passengers (those waiting at B at that time). It then returns to A bringing those passengers. The whole trip (from A to B and back) takes an average amount of time T.

Passengers arrive randomly at each location ℓ at a rate $\lambda_\ell, \ell = A, B$, and each passenger wants to go to the other location. For simplicity, assume that the bus has been designed large enough to accommodate all the passengers that may travel between the two locations.

Decision epochs always occur with the bus stationed in A, whenever a passenger arrives at either location, or immediately after the bus arrives at A after a roundtrip. At each decision epoch, the dispatcher must decide whether to dispatch the bus (S) or to wait for the next passenger to arrive at either location (W). If any of the two locations is "full" (i.e., the number of passengers waiting at that location is N), the dispatcher must necessarily select action S.

The decision process of the dispatcher can be modeled as an MDP as follows.

- The state $\mathbf{x}_t$ at time step t is given by $\mathbf{x}_t = (\mathrm{x}_{a,t}, \mathrm{x}_{b,t})$, where $\mathrm{x}_{\ell,t}$ is the number of passengers waiting at location ℓ at time step t. We have that $\mathcal{X} = \{0, \ldots, N\}^2$.

- The action space is $\mathcal{A} = \{S, W\}$.

- As for the transition probabilities, we first consider that the bus is stationed at A (i.e., the last action of the dispatcher was W). The probability that the next passenger arrives at location A (and not B) is $\frac{\lambda_A}{\lambda_A + \lambda_B}$; conversely, the

probability that the next passenger arrives at location B is $\frac{\lambda_B}{\lambda_A+\lambda_B}$. Then

$$\boldsymbol{P}((k,\ell) \mid (m,n), W) = \begin{cases} \frac{\lambda_A}{\lambda_A+\lambda_B} & \text{if } k = m \text{ and } \ell = n+1 \\ \frac{\lambda_B}{\lambda_A+\lambda_B} & \text{if } k = m+1 \text{ and } \ell = n \\ 0 & \text{otherwise.} \end{cases}$$

When the bus is dispatched, all passengers waiting in both locations A and B are served. However, new passengers may arrive when the bus is in transit. New arrivals at A follow a Poisson distribution with parameter $\lambda_A D$, corresponding to the rate of arrival at A times the time of the roundtrip. Similarly, the probability of arrival at B follows a Poisson distribution with parameter $\lambda_B D/2$. Finally, arrivals at A and B are independent, yielding

$$\boldsymbol{P}((k,\ell) \mid (m,n), S) = \mathsf{Poisson}(k; \lambda_A D) \times \mathsf{Poisson}(\ell; \lambda_B D/2).$$

- The cost incurred by the dispatcher is proportional to the amount of time that each passenger waits (in average) at the corresponding stop. For $\boldsymbol{x} = (m,n)$,

$$c(\boldsymbol{x}, W) = K_{\text{wait}} \frac{m+n}{\lambda_A + \lambda_B}.$$

When the bus is dispatched, let K_{round} denote the the cost of a roundtrip. Additionally, the passengers at stop B wait $D/2$ before the bus gets there. During the round trip, passengers arrive at location A at a rate λ_A and wait from their arrival time until the bus arrives back at A. This corresponds to an average time of $\frac{1}{2}\lambda_A D^2$. Similarly, when the bus leaves location B, passengers that arrive in the meantime will wait, in average $\frac{1}{4}\lambda_B D^2$. Putting all together, we get

$$\begin{aligned} c(\boldsymbol{x}, S) &= K_{\text{round}} + K_{\text{wait}} \left[n\frac{D}{2} + \lambda_A \frac{D^2}{2} + \lambda_B \frac{D^2}{4} \right] \\ &= K_{\text{round}} + K_{\text{wait}} \frac{D^2}{2} \left[\frac{n}{D} + \lambda_A + \frac{\lambda_B}{2} \right] \end{aligned}$$

for some constants K_{wait} and K_{round}.

5.3 Optimality ($\star$)

This section introduces the fundamental result in the theory of Markov decision problems: given a finite MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$, with $\gamma \in [0,1)$, we show that there is an optimal policy that is both deterministic and stationary. Additionally, the proof of existence of such optimal policy lays the groundwork for the computational solutions of MDPs.

5.3.1 Properties of cost-to-go functions

Recall that, given an MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$, an optimal policy is any policy π^* such that, *simultaneously for all states* $x \in \mathcal{X}$,

$$J^{\pi^*}(x) \leq J^{\pi}(x).$$

Solving the Markov decision problem $\mathcal{M}$ consists in determining whether any such a policy exists for $\mathcal{M}$ and, if so, finding it. The optimal policy exists as long as the supremum in (5.4) is attained. In other words, letting

$$J^*(x) = \inf_{\pi \in \Pi^{\mathrm{HR}}} J^\pi(x), \tag{5.5}$$

an optimal policy exists as long as there is $\pi^* \in \Pi^{\mathrm{HR}}$ such that

$$J^{\pi^*}(x) = J^*(x),$$

for all $x \in \mathcal{X}$. We start with the following convenient fact.

Proposition 5.1. *Given a finite MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$ and a policy $\pi \in \Pi^{\mathrm{HR}}$, then for any initial state $x \in \mathcal{X}$ there is a policy $\pi' \in \Pi^{\mathrm{MR}}$ (which possibly depends on x) such that*

$$J^\pi(x) = J^{\pi'}(x).$$

Proof. See Section 5.8. □

Proposition 5.1 means that, for any given initial state, we can focus our search for optimal policies on the set Π^{MR}. We thus start our path towards establishing the existence of optimal policies by looking into the properties of cost-to-go functions associated with Markov policies.

The operator T_π

As seen in Section 5.2.1, any policy $\pi \in \Pi^{\mathrm{MR}}$ induces a Markov cost process $\{(\mathrm{x}_t, \mathrm{c}_t), t \in \mathbb{N}\}$. The associated cost-to-go function verifies

$$\begin{aligned} J^\pi(x_0) &= \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathrm{c}_t \mid \mathrm{x}_0 = x_0 \right] \\ &= \mathbb{E}_\pi \left[\mathrm{c}_0 + \sum_{t=1}^{\infty} \gamma^t \mathrm{c}_t \mid \mathrm{x}_0 = x_0 \right] \\ &= \mathbb{E}_\pi \left[\mathrm{c}_0 \mid \mathrm{x}_0 = x_0 \right] + \gamma \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathrm{c}_{t+1} \mid \mathrm{x}_0 = x_0 \right]. \end{aligned} \tag{5.6}$$

The second term in (5.6) resembles a cost-to-go, but referring to the process for $t \geq 1$ and conditioned on the value of x_0. However, using the total probability law (see Appendix A),

$$\begin{aligned} &J^\pi(x_0) \\ &= \mathbb{E}_\pi \left[\mathrm{c}_0 \mid \mathrm{x}_0 = x_0 \right] + \gamma \sum_{y \in \mathcal{X}} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathrm{c}_{t+1} \mid \mathrm{x}_1 = y, \mathrm{x}_0 = x_0 \right] \mathbb{P}_\pi \left[\mathrm{x}_1 = y \mid \mathrm{x}_0 = x_0 \right] \\ &= \mathbb{E}_\pi \left[\mathrm{c}_0 \mid \mathrm{x}_0 = x_0 \right] + \gamma \sum_{y \in \mathcal{X}} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathrm{c}_{t+1} \mid \mathrm{x}_1 = y \right] \mathbb{P}_\pi \left[\mathrm{x}_1 = y \mid \mathrm{x}_0 = x_0 \right], \end{aligned} \tag{5.7}$$

where the second equality follows from the Markov property of $\{(\mathrm{x}_t, \mathrm{c}_t), t \in \mathbb{N}\}$. Let us carefully consider (5.7). Using the total probability law, the first term becomes

$$\begin{aligned}\mathbb{E}_\pi\left[\mathrm{c}_0 \mid \mathrm{x}_0 = x_0\right] &= \sum_{a \in \mathcal{A}} \mathbb{E}_\pi\left[\mathrm{c}_0 \mid \mathrm{x}_0 = x_0, \mathrm{a}_0 = a\right] \mathbb{P}_\pi\left[\mathrm{a}_0 = a \mid \mathrm{x}_0 = x_0\right] \\ &= \sum_{a \in \mathcal{A}} c(x_0, a)\pi_0(a \mid x_0).\end{aligned}$$

The expectation inside the summation corresponds to the cost-to-go associated with the Markov policy resulting from *advancing* π one step in time. In other words, the expectation corresponds to the value $J^{\pi^+}(y)$, where π^+ is the Markov policy defined as

$$\pi_t^+(a \mid x) = \pi_{t+1}(a \mid x),$$

for all $x \in \mathcal{X}, a \in \mathcal{A}$ and $t \in \mathbb{N}$.

Finally, $\mathbb{P}_\pi\left[\mathrm{x}_1 = y \mid \mathrm{x}_0 = x_0\right]$ verifies

$$\begin{aligned}\mathbb{P}_\pi\left[\mathrm{x}_1 = y \mid \mathrm{x}_0 = x_0\right] &= \sum_{a \in \mathcal{A}} \mathbb{P}_\pi\left[\mathrm{x}_1 = y \mid \mathrm{x}_0 = x_0, \mathrm{a}_0 = a\right] \mathbb{P}_\pi\left[\mathrm{a}_0 = a \mid \mathrm{x}_0 = x_0\right] \\ &= \sum_{a \in \mathcal{A}} \boldsymbol{P}(y \mid x_0, a)\pi_0(a \mid x_0).\end{aligned}$$

Putting everything together, we finally get

$$J^\pi(x_0) = \sum_{a \in \mathcal{A}} \pi_0(a \mid x_0) \left[c(x_0, a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x_0, a) J^{\pi^+}(y) \right]. \tag{5.8}$$

Equation 8.14 evidences the impact of the decision at time step $t = 0$ in the cost-to-go of policy π. In the particular case where π is stationary, $\pi^+ = \pi$ and we get

$$J^\pi(x_0) = c_\pi(x_0) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}_\pi(y \mid x_0) J^\pi(y), \tag{5.9}$$

where c_π and $\boldsymbol{P}_\pi$ are defined in Section 5.2.1. Given an arbitrary function $J : \mathcal{X} \to \mathbb{R}$, define the operator T_π component-wise as

$$(\mathsf{T}_\pi J)(x) = c_\pi(x) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}_\pi(y \mid x) J(y). \tag{5.10}$$

Then, (5.9) can be written equivalently as

$$J^\pi = \mathsf{T}_\pi J^\pi, \tag{5.11}$$

i.e., J^π is the *fixed-point* of operator T_π. We have the following result.

Proposition 5.2. *Given an MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$ and a stationary policy π, the cost-to-go function J^π is the only fixed point of the operator T_π.*

Proof. See Section 5.8. □

We can also write (5.9) in matrix form as

$$\boldsymbol{J}^\pi = \boldsymbol{c}_\pi + \gamma \boldsymbol{P}_\pi \boldsymbol{J}^\pi. \tag{5.12}$$

Since $\boldsymbol{P}_\pi$ is a stochastic matrix, its eigenvalues are all smaller than or equal to 1 (see Appendix A), implying that the matrix $(\boldsymbol{I} - \gamma \boldsymbol{P}_\pi)$ is invertible. Therefore J^π can be computed as

$$\boldsymbol{J}^\pi = (\boldsymbol{I} - \gamma \boldsymbol{P}_\pi)^{-1} \boldsymbol{c}_\pi.$$

The Bellman optimality equation

If we combine (8.14) with (5.5), we get that

$$J^*(x_0) = \inf_{\pi \in \Pi^{\mathrm{MR}}} \sum_{a \in \mathcal{A}} \pi_0(a \mid x_0) \left[c(x_0, a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x_0, a) J^{\pi^+}(y) \right].$$

The decision at time step $t = 0$ affects the cost-to-go for $t > 1$ only through the resulting state y. In other words, the value $J^{\pi^+}(y)$ does not depend directly on π_0, and we can write

$$\begin{aligned} J^*(x_0) &= \inf_{\pi_0 \in \Delta(\mathcal{A})} \inf_{\pi^+ \in \Pi^{\mathrm{MR}}} \sum_{a \in \mathcal{A}} \pi_0(a \mid x_0) \left[c(x_0, a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x_0, a) J^{\pi^+}(y) \right] \\ &= \inf_{\pi_0 \in \Delta(\mathcal{A})} \sum_{a \in \mathcal{A}} \pi_0(a \mid x_0) \left[c(x_0, a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x_0, a) J^*(y) \right] \end{aligned} \tag{5.13}$$

Using Lemma 4.6 in Chapter 4 (see page 92),

$$\begin{aligned} J^*(x_0) &= \inf_{\pi_0 \in \Delta(\mathcal{A})} \sum_{a \in \mathcal{A}} \pi_0(a \mid x_0) \left[c(x_0, a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x_0, a) J^*(y) \right] \\ &= \min_{a \in \mathcal{A}} \left[c(x_0, a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x_0, a) J^*(y) \right]. \end{aligned}$$

Two important observations are in order. First, the derivations above suggest that—if J^* is known—the optimal decision at time step $t = 0$ can be computed as

$$\pi_0^*(x_0) = \operatorname*{argmin}_{a \in \mathcal{A}} \left[c(x_0, a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x_0, a) J^*(y) \right], \tag{5.14}$$

since it is the decision that attains the infimum value at x_0. Second, since x_0 is arbitrary, we showed that J^* verifies the recursive relation

$$J^*(x) = \min_{a \in \mathcal{A}} \left[c(x,a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x, a) J^*(y) \right] \tag{5.15}$$

for any $x \in \mathcal{X}$. The recursion (5.15) is known as the *Bellman optimality equation* and lies at the core of most MDP solution methods. Given an arbitrary function $J : \mathcal{X} \to \mathbb{R}$, define the operator T component-wise as

$$(\mathsf{T}J)(x) = \min_{a \in \mathcal{A}} \left[c(x,a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x, a) J(y) \right]. \tag{5.16}$$

The following result is the counterpart to Proposition 5.2 for the operator T.

Proposition 5.3. *Given an MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$, the cost-to-go function J^* is the only fixed point of the operator T.*

Proof. See Section 5.8. □

Proposition 5.3 yields two important implications:

- It guarantees that the infimum in (5.5) is the fixed point of operator T. It exists and is unique.
- Moreover, (5.14) suggests the deterministic stationary policy π^* given by

$$\pi^*(x) = \operatorname*{argmin}_{a \in \mathcal{A}} \left[c(x,a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x, a) J^*(y) \right] \tag{5.17}$$

  as a possible candidate to optimal policy.

In the continuation, we introduce the main result of this chapter. We show that the policy π^* in (5.17) is indeed optimal, since $J^{\pi^*} = \mathsf{T}J^{\pi^*}$. This establishes the existence of optimal deterministic and stationary policies for finite MDPs, and constitutes the main result of this chapter.

5.3.2 Existence of optimal policies

Consider a bounded function $J : \mathcal{X} \to \mathbb{R}$. We say that a (deterministic) policy π_g is *greedy* w.r.t. J if

$$\pi_g(x) = \operatorname*{argmin}_{a \in \mathcal{A}} \left[c(x,a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x, a) J(y) \right],$$

for all $x \in \mathcal{X}$. In other words, a policy π_g is greedy w.r.t. J if

$$\mathsf{T}_{\pi_g} J = \mathsf{T} J.$$

The policy π^* defined in (5.17) is, therefore, greedy w.r.t. J^*. By construction,

$$J^* = \mathsf{T}_{\pi^*} J^* = \mathsf{T} J^*.$$

On the other hand, by Proposition 5.2, J^{π^*} is the only fixed point of T_{π^*}, which means that $J^* = J^{\pi^*}$ and π^* is indeed an optimal policy.

We proved the following central result.

Theorem 5.4. *Given an MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$, there is an optimal deterministic stationary policy π^* given, for each $x \in \mathcal{X}$, by*

$$\pi^*(x) \in \operatorname*{argmin}_{a \in \mathcal{A}} \left[c(x,a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x, a) J^*(y) \right],$$

where J^* is the fixed point of T.

Since the function J^* is the cost-to-go function associated with the optimal policy, we henceforth refer to the cost-to-go function J^* as the *optimal cost-to-go function*.

Although our presentation assumes finite $\mathcal{X}$ and $\mathcal{A}$, Theorem 5.4 extends essentially unmodified to more general settings—for example when $\mathcal{X}$ is countable. We briefly discuss the more general situation in Section 5.5. Additional details can also be found in the bibliographical notes in Section 5.7.

We conclude this section by stating the principle translated by (5.17), known as the *Bellman optimality principle*.

Bellman optimality principle (Bellman, 1954)

An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions.

5.4 MDP solution methods

Within the framework of Markov decision problems, we can identify two distinct computational challenges:

- Given an MDP and a fixed stationary policy π, compute the cost-to-go function J^π associated with the induced Markov cost process. Such computation is sometimes referred as the (policy) *evaluation problem* or the *prediction problem* in MDPs.
- Given an MDP, compute the associated optimal policy π^*. Such computation is sometimes referred as the (policy) *optimization problem* or the *control problem* in MDPs.

We survey 3 main classes of methods, broadly known as *value iteration*, *policy iteration* and *linear programming*.

5.4.1 Value iteration

Value iteration (VI) is a class of dynamic programming approaches that iteratively compute cost-to-go functions using the operators T_π and T introduced in Section 5.3.1.

Prediction

Let $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$ denote an MDP and π a fixed stationary policy. As seen in Section 5.3.1, the cost-to-go function J^π is the solution to the linear system of equations (5.12) and can be computed directly as

$$\boldsymbol{J}^\pi = (\boldsymbol{I} - \gamma \boldsymbol{P}_\pi)^{-1} \boldsymbol{c}_\pi. \tag{5.18}$$

However, for systems in which the matrix inversion in (5.18) is computationally too expensive, an iterative approach may be preferable. Taking advantage of the fact that the operator T_π is a *contraction*[4] and has J^π as its (unique) fixed point (see Proposition 5.2), it is possible to construct a sequence $\left\{J^{(k)}, k \in \mathbb{N}\right\}$ that converges to J^π as follows.

Given an initial estimate $J^{(0)}$, let $J^{(k+1)} = \mathsf{T}_\pi J^{(k)}$ or, explicitly,

$$\begin{aligned} J^{(k+1)}(x) &= c_\pi(x) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}_\pi(y \mid x) J^{(k)}(y) \\ &= \sum_{a \in \mathcal{A}} \pi(a \mid x) \left[c(x,a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x, a) J^{(k)}(y) \right], \end{aligned}$$

for all $x \in \mathcal{X}$. The sequence thus constructed converges to J^π, and leads to Algorithm 5.1.

The following result is a corollary of Proposition 5.2, and ensures that Algorithm 5.1 indeed converges to the desired cost-to-go function.

[4] A mapping $F : \mathcal{X} \to \mathcal{X}$ is a contraction in $\mathcal{X}$ if $\|F(x_1) - F(x_2)\| \leq \gamma \|x_1 - x_2\|$, for any $x_1, x_2 \in \mathcal{X}$ (see Appendix A).

Algorithm 5.1 VI to compute J^π.

Require: MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$; tolerance $\varepsilon > 0$; policy π;
1: Initialize $k = 0$, $J^{(0)} \equiv 0$
2: **repeat**
3: $\quad J^{(k+1)} \leftarrow \mathsf{T}_\pi J^{(k)}$
4: $\quad k \leftarrow k + 1$
5: **until** $\left\|J^{(k-1)} - J^{(k)}\right\|_2 < \varepsilon$.
6: **return** $J^{(k)}$

Corollary 5.5. *Given an MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$ and a stationary policy π, the sequence $\left\{J^{(k)}, k \in \mathbb{N}\right\}$ generated by Algorithm 5.1 converges to J^π, i.e.,*

$$\lim_{k\to\infty} J^{(k)} = J^\pi.$$

The stopping condition in line 5 can be selected to ensure any desired error for $J^{(k)}$. If, for example, we want to ensure that $\left\|J^{(k)} - J^\pi\right\|_2 < \delta$, for some given error δ, we have

$$\begin{aligned}
\left\|J^{(k)} - J^\pi\right\|_2 &= \left\|J^{(k)} - J^{(k+1)} + J^{(k+1)} - J^\pi\right\|_2 \\
&\le \left\|J^{(k)} - J^{(k+1)}\right\|_2 + \left\|J^{(k+1)} - J^\pi\right\|_2 \\
&= \left\|\mathsf{T}_\pi J^{(k-1)} - \mathsf{T}_\pi J^{(k)}\right\|_2 + \left\|\mathsf{T}_\pi J^{(k)} - \mathsf{T}_\pi J^\pi\right\|_2 \\
&\le \gamma\left\|J^{(k-1)} - J^{(k)}\right\|_2 + \gamma\left\|J^{(k)} - J^\pi\right\|_2,
\end{aligned}$$

where the last inequality follows from the fact that T_π is a contraction. Therefore,

$$\left\|J^{(k)} - J^\pi\right\|_2 \le \frac{\gamma}{1-\gamma}\left\|J^{(k-1)} - J^{(k)}\right\|_2, \tag{5.19}$$

and the desired error can be obtained by setting $\varepsilon < \dfrac{\delta}{\gamma}(1-\gamma)$.

Another observation is that line 3 updates every component of $J^{(k)}$ simultaneously. However, it is also possible to use the most up-to-date values of $J^{(k+1)}$ in updating each component of $J^{(k)}$. Assuming without loss of generality that the states in $\mathcal{X}$ are ordered as $x_1, x_2, \ldots, x_{|\mathcal{X}|}$, we can replace the vector update in line 3 by a component-wise update

$$\begin{aligned}
&J^{(k+1)}(x_n) \\
&= c_\pi(x_n) + \gamma \sum_{m<n} \boldsymbol{P}_\pi(x_m \mid x_n) J^{(k+1)}(x_m) + \gamma \sum_{m\ge n} \boldsymbol{P}_\pi(x_m \mid x_n) J^{(k)}(x_m),
\end{aligned} \tag{5.20}$$

where the components $J^{(k+1)}(x_n)$ are update sequentially according to the order of $\mathcal{X}$. This variation of Algorithm 5.1 is known as *Gauss-Seidel value iteration*, and is more efficient than the version presented in 5.1, as established in the following result.

Proposition 5.6. *Given an MDP $(\mathcal{X}, \mathcal{A}\ \{\boldsymbol{P}_a\}, c, \gamma)$ and a stationary policy π, the sequence $J^{(k)}$ generated by Algorithm 5.1 with update rule (5.20) converges to J^π at a rate no larger than γ (and possibly smaller).*

Proof. See Section 5.8. □

Control

We now discuss how to actually solve a Markov decision problem, i.e., how to compute the optimal policy π^* that dictates how the agent should act to minimize the expected discounted cost. To this purpose we rely on the fundamental result stated in Theorem 5.4: given J^*, the optimal policy can be computed as

$$\pi^*(x) \in \operatorname*{argmin}_{a \in \mathcal{A}} \left[c(x,a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x, a) J^*(y) \right] \tag{5.21}$$

for all $x \in \mathcal{X}$. Although J^* cannot be computed as the solution to a linear system, it can be computed iteratively in a manner essentially similar to the one used to compute J^π. Taking advantage of the fact that T is a contraction, we again generate a sequence $\left\{J^{(k)}, k \in \mathbb{N}\right\}$ that converges to J^*.

The resulting algorithm is also an instance of VI, and is summarized in Algorithm 5.2.

Algorithm 5.2 VI to compute J^*.

Require: MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$; tolerance $\varepsilon > 0$;
1: Initialize $k = 0$, $J^{(0)} \equiv 0$
2: **repeat**
3: $\quad J^{(k+1)} \leftarrow \mathsf{T} J^{(k)}$
4: $\quad k \leftarrow k + 1$
5: **until** $\left\| J^{(k-1)} - J^{(k)} \right\|_\infty < \varepsilon$.
6: Compute π^* using (5.21)
7: **return** π^*

Component-wise, the update in line 3 is equivalent to

$$J^{(k+1)}(x) = \min_{a \in \mathcal{A}} \left[c(x,a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x, a) J^{(k)}(y) \right].$$

Therefore, except for the operator behind each of the Algorithms 5.1 and 5.2, the two are essentially similar. Therefore, in much the same way that Algorithm 5.1 converges to J^π, Algorithm 5.2 converges to J^*, as stated in the next corollary.

Corollary 5.7. *Given an MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$, the sequence $\left\{J^{(k)}, k \in \mathbb{N}\right\}$ gen-*

erated by Algorithm 5.2 converges to J^*, i.e.,

$$\lim_{k\to\infty} J^{(k)} = J^*.$$

Additionally, the Gauss-Seidel modification can also be applied without change to Algorithm 5.2 while retaining the same convergence guarantees. There is, however, one important distinction between the two algorithms, regarding the stopping condition.

The computation of J^* serves as an intermediate step to compute π^*. Therefore, instead of requiring that $\left\|J^{(k)} - J^*\right\| < \delta$, it is natural to require that $\left\|J^{\pi_g^{(k)}} - J^*\right\| < \delta$, where $\pi_g^{(k)}$ denotes the greedy policy w.r.t. $J^{(k)}$.

Repeating the steps used to derive (5.19), we get that

$$\left\|J^{(k)} - J^*\right\|_\infty \leq \frac{\gamma}{1-\gamma}\left\|J^{(k-1)} - J^{(k)}\right\|_\infty$$

and

$$\left\|J^{\pi_g^{(k)}} - J^{(k)}\right\|_\infty \leq \frac{\gamma}{1-\gamma}\left\|J^{(k-1)} - J^{(k)}\right\|_\infty.$$

Therefore, since

$$\left\|J^{\pi_g^{(k)}} - J^*\right\|_\infty \leq \left\|J^{\pi_g^{(k)}} - J^{(k)}\right\|_\infty + \left\|J^{(k)} - J^*\right\|$$

we finally get the bound

$$\left\|J^{\pi_g^{(k)}} - J^*\right\|_\infty \leq \frac{2\gamma}{1-\gamma}\left\|J^{(k-1)} - J^{(k)}\right\|_\infty. \tag{5.22}$$

The desired error can thus be obtained by setting $\varepsilon < \dfrac{\delta}{2\gamma}(1-\gamma)$.

Alternative value functions

Previously, we noted that the optimal cost-to-go function J^* can be used to compute the optimal policy as in (5.21). Such computation requires the whole MDP model, since the cost function and transition probabilities are featured prominently in (5.21).

Alternatively, and following the notation in Chapter 4, we revisit the notion of *action-value function* or *Q-function*, which maps each state-action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$ to a cost-to-go. Formally, given an MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$, the Q-function associated with a policy π is the function $Q^\pi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ defined as

$$Q^\pi(x, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t \mathrm{c}_t \mid \mathrm{x}_0 = x, \mathrm{a}_0 = a\right].$$

The Q-function associated with the optimal policy, π^*, is called the *optimal Q-function*, and denoted by Q^*.

Q-functions are, conceptually, "enhanced" cost-to-go functions that allow, at each state, a direct comparison between the different actions in terms of their

contribution to the expected cost-to-go. Therefore, it is not surprising that there is a close relation between a Q-function and the associated cost-to-go function. In particular, for any stationary policy π, we can compute Q^π from J^π as

$$Q^\pi(x,a) = c(x,a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x,a) J^\pi(y). \tag{5.23}$$

Conversely, the cost-to-go function can also be computed from the corresponding Q-function as

$$J^\pi(x) = \sum_{a \in \mathcal{A}} \pi(a \mid x) Q^\pi(x,a). \tag{5.24}$$

For the particular case of J^*, we have

$$J^*(x) = \min_{a \in \mathcal{A}} Q^*(x,a) \tag{5.25}$$

and it follows that

$$\pi^*(x) = \operatorname*{argmin}_{a \in \mathcal{A}} Q^*(x,a). \tag{5.26}$$

The relation with the expected utility framework of Chapter 4 becomes once again evident by revisiting (4.7) and noting the similarity with (5.26). Additionally, as evidenced in (5.26), the optimal policy can be computed directly from Q^*, which will prove useful when we address situations in which the cost function, c, and the transition probabilities, $\{\boldsymbol{P}_a\}$, are not available.

Q-functions are also interesting from a computational point of view, since Algorithms 5.1 and 5.2 admit direct generalizations to compute Q-functions. Replacing (5.24) and (5.25) in (5.23) yields

$$Q^\pi(x,a) = c(x,a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x,a) \sum_{a' \in \mathcal{A}} \pi(a' \mid y) Q^\pi(y,a') \tag{5.27}$$

and

$$Q^*(x,a) = c(x,a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x,a) \min_{a' \in \mathcal{A}} Q^*(y,a'). \tag{5.28}$$

From (5.27) and (5.28) we define two new operators, H_π and H, as follows. Given an arbitrary function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, let

$$(\mathsf{H}_\pi Q)(x,a) = c(x,a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x,a) \sum_{a' \in \mathcal{A}} \pi(a' \mid y) Q(y,a'),$$

$$(\mathsf{H} Q)(x,a) = c(x,a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x,a) \min_{a' \in \mathcal{A}} Q^*(y,a').$$

Both H_π and H are contractions that have Q^π and Q^* as unique fixed points (see Exercise **??**). They can be used to generate sequences $\left\{Q^{(k)}, k \in \mathbb{N}\right\}$ that converge to Q^π or Q^* simply by replacing T_π and T by H_π and H in Algorithms 5.1 and 5.2. The two resulting algorithms are also value iteration algorithms and retain the same convergence guarantees. For example, the algorithm to compute Q^* is summarized in Algorithm 5.3.

Algorithm 5.3 VI to compute Q^*.

Require: MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$; tolerance $\varepsilon > 0$;
1: Initialize $k = 0$, $Q^{(0)} \equiv 0$
2: **repeat**
3: $\quad Q^{(k+1)} \leftarrow \mathsf{H}Q^{(k)}$
4: $\quad k \leftarrow k + 1$
5: **until** $\left\|Q^{(k-1)} - Q^{(k)}\right\|_\infty < \varepsilon$.
6: Compute π^* using (5.26)
7: **return** π^*

$\diamond$

It is possible to go even further in reducing an MDP to the framework of Chapter 4. In particular, it is possible to single out, in each state $x \in \mathcal{X}$, how each individual action differs from a prescribed action. Given an MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$, the *advantage function* associated with a policy π is the function $A^\pi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ defined as

$$A^\pi(x, a) = J^\pi(x) - Q^\pi(x, a).$$

The usefulness of the advantage function becomes apparent in the case of the optimal policy: since $J^*(x) \leq Q^*(x, a)$ for all $a \in \mathcal{A}$, $A^*(x, a) \leq 0$ which means that any action (other than the optimal actions) offers a negative advantage in terms of cost-to-go. For a general policy π, if $A^\pi(x, a) > 0$ then action a may potentially lead to improved cost-to-go. Unfortunately, the advantage function does not verify a recursive relation and is, therefore, difficult to compute directly and seldom used in practice.

5.4.2 Policy iteration

The methods discussed in Section 5.4.1 to compute the optimal policy are focused on the computation of the cost-to-go functions J^* or Q^*. These methods take advantage of the recursive nature of J^* and Q^* to iteratively compute these functions. However, the convergence guarantees for value iteration are asymptotic, meaning that convergence to the actual cost-to-go function takes place only as the number of iterations $k \to \infty$.

On the other hand, if $\mathcal{X}$ and $\mathcal{A}$ are finite, the number of possible (deterministic and stationary) policies is finite—there are as many as $|\mathcal{A}|^{|\mathcal{X}|}$ different policies. Therefore, even a naive brute-force search will guarantee that the optimal policy is computed in a finite number of iterations.

In practice, the argument above is somewhat misleading, as value iteration usually conveys fairly quickly a good approximation to J^* or Q^*, from which the optimal policy can be extracted. Nevertheless, the possibility of efficiently searching the space of deterministic stationary policies for the optimal policy is worth exploring.

Policy iteration (PI) iteratively searches the space of policies by constructing a sequence $\left\{\pi^{(k)}, k \in \mathbb{N}\right\}$, where each policy $\pi^{(k+1)}$ is better than $\pi^{(k)}$ in terms of the resulting cost-to-go. Since the number of policies is finite, such improvement guarantee ensures that, in a finite number of iterations (usually very few) the optimal policy is discovered.

The fundamental result that supports policy iteration is summarized in Proposition 5.8. Policy iteration is summarized in Algorithm 5.4.

Proposition 5.8. *Given an MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$, let J^π denote the cost-to-go function associated with some deterministic stationary policy π. If π_g is the greedy policy w.r.t. J^π, then*

$$J^{\pi_g}(x) \leq J^\pi(x),$$

for every $x \in \mathcal{X}$. Moreover, if $J^{\pi_g}(x) = J^\pi(x)$ for every $x \in \mathcal{X}$, then $J^{\pi_g} = J^*$ and $\pi_g = \pi^*$.

Proof. See Section 5.8. □

Algorithm 5.4 Policy iteration to compute π^*.

Require: MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$
1: Initialize $k = 0$, $\pi^{(0)}$ randomly
2: **repeat**
3: $\quad \boldsymbol{J} \leftarrow (\boldsymbol{I} - \gamma \boldsymbol{P}_{\pi^{(k)}})^{-1} \boldsymbol{c}_{\pi^{(k)}}$
4: $\quad \pi^{(k+1)} \leftarrow \pi_g^J$
5: $\quad k \leftarrow k + 1$
6: **until** $\pi^{(k-1)} = \pi^{(k)}$
7: **return** $\pi^{(k)}$

Each iteration of the policy iteration algorithm can be broken down in two steps: an *evaluation step* (line 3), where the cost-to-go associated with the current policy is estimated; and an *improvement step* (line 4), where the cost-to-go function is used to improve the policy. That PI actually delivers π^* is guaranteed by the following corollary of Proposition 5.8.

Corollary 5.9. *Given an MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$ and any initial policy $\pi^{(0)}$, the sequence of policies $\left\{\pi^{(k)}, k \in \mathbb{N}\right\}$ generated by Algorithm 5.4 converges to π^* in a finite number of steps.*

In practice, PI is a very efficient algorithm, computing the optimal policy in a small number of iterations. However, for large systems, the matrix inversion in the policy evaluation step may be computationally too expensive. Alternatively, one could use Algorithm 5.1 to compute J^π. In fact, it is not necessary to run VI to completion at each iteration of PI, but only a small number of policy evaluation steps between any two policy improvement steps. This modified version of PI is summarized in Algorithm 5.5.

Algorithm 5.5 Policy iteration with iterative policy evaluation.

Require: MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$; number of VI iterations, M

Initialize $k = 0$; $J \equiv 0$; $\pi^{(0)}$ randomly
repeat
  for $m = 1 : M$ **do**
    $J \leftarrow \mathsf{T}_{\pi^{(k)}} J$
  end for
  $\pi^{(k+1)} \leftarrow \pi_g^J$
  $k \leftarrow k + 1$
until $\pi^{(k-1)} = \pi^{(k)}$
return $\pi^{(k)}$

5.4.3 Linear programming ($\star$)

Value and policy iteration methods rely, at a very fundamental level, on the computation of cost-to-go functions. Value iteration computes J^* (or Q^*) from which the optimal policy can then be computed. Policy iteration, on the other hand, builds a sequence of policies converging to π^*; for each policy π in the sequence, the corresponding cost-to-go function, J^π, is computed.

An alternative approach to solving Markov decision problems—particularly useful when approximations must be used—uses *linear programming* to compute J^*. As seen in Section 5.3.1, J^* is the unique fixed point of the operator T. We establish this fact in Proposition 5.3, where we use another useful property of the operator T: given a bounded function $J : \mathcal{X} \to \mathbb{R}$, if $J(x) \leq (\mathsf{T}J)(x)$ for any $x \in \mathcal{X}$, then $J(x) \leq J^*(x)$ (see Section 5.8 for details).

In other words, J^* is (component-wise) the largest function $J : \mathcal{X} \to \mathbb{R}$ that verifies the restriction $J(x) \leq (\mathsf{T}J)(x)$. Let μ denote an arbitrary distribution over $\mathcal{X}$, with $\mu(x) > 0$ for all $x \in \mathcal{X}$. Then, J^* is the solution to the linear program

$$\begin{aligned} \text{maximize} \quad & \sum_{x \in \mathcal{X}} \mu(x) J(x) \\ \text{subject to} \quad & J(x) - \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x, a) J(y) \leq c(x, a), \forall x \in \mathcal{X}, a \in \mathcal{A}. \end{aligned} \tag{5.29}$$

The formulation in (5.29) differs from that in Appendix A in that we did not use vector notation, since the more explicit, component-based formulation is more convenient for analysis. Using the same component-wise formulation, the dual of (5.29) is

$$\begin{aligned} \text{minimize} \quad & \sum_{x \in \mathcal{X}, a \in \mathcal{A}} \eta(x, a) c(x, a) \\ \text{subject to} \quad & \sum_{x \in \mathcal{X}, a \in \mathcal{A}} \eta(x, a)(1 - \boldsymbol{P}(y \mid x, a)) \geq \mu(y), \forall y \in \mathcal{X} \\ & \eta(x, a) \geq 0, \forall x \in \mathcal{X}, a \in \mathcal{A}. \end{aligned} \tag{5.30}$$

Since J^* is the optimal solution to the primal in (5.29), it follows that the dual in

(5.30) also has a solution and both solutions have the same value, given by

$$v = \sum_{x \in \mathcal{X}} \mu(x) J^*(x).$$

If we take μ as the initial distribution for the MDP, the value v corresponds to the expected optimal cost-to-go when the initial state is distributed according to μ. Formally,

$$v = \sum_{x \in \mathcal{X}} \mu(x) J^*(x) = \sum_{x \in \mathcal{X}} \mathbb{P}\left[\mathrm{x}_0 = x\right] \mathbb{E}_{\pi^*}\left[\sum_{t=0}^{\infty} \gamma^t \mathrm{c}_t \mid \mathrm{x}_0 = x\right].$$

Explicitly writing the expectation over trajectories yields

$$\begin{aligned} v &= \sum_{x \in \mathcal{X}} \mathbb{P}\left[\mathrm{x}_0 = x\right] \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\pi^*}\left[\mathrm{c}_t \mid \mathrm{x}_0 = x\right] \\ &= \sum_{x \in \mathcal{X}} \mathbb{P}\left[\mathrm{x}_0 = x\right] \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\pi^*}\left[c(\mathrm{x}_t, \mathrm{a}_t) \mid \mathrm{x}_0 = x\right] \\ &= \sum_{x \in \mathcal{X}} \mathbb{P}\left[\mathrm{x}_0 = x\right] \sum_{t=0}^{\infty} \gamma^t \sum_{x' \in \mathcal{X}, a \in \mathcal{A}} c(x', a) \mathbb{P}_{\pi^*}\left[\mathrm{x}_t = x', \mathrm{a}_t = a \mid \mathrm{x}_0 = x\right]. \end{aligned}$$

Finally, rearranging the summations—and replacing by the definition of v in the dual—we get

$$\begin{aligned} v &= \sum_{x' \in \mathcal{X}, a \in \mathcal{A}} \sum_{x \in \mathcal{X}} \mathbb{P}\left[\mathrm{x}_0 = x\right] \sum_{t=0}^{\infty} \gamma^t \mathbb{P}_{\pi^*}\left[\mathrm{x}_t = x', \mathrm{a}_t = a \mid \mathrm{x}_0 = x\right] c(x', a) \\ &= \sum_{x' \in \mathcal{X}, a \in \mathcal{A}} \eta(x', a) c(x', a). \end{aligned}$$

We conclude that

$$\eta(x, a) = \sum_{x' \in \mathcal{X}} \mathbb{P}\left[\mathrm{x}_0 = x'\right] \sum_{t=0}^{\infty} \gamma^t \mathbb{P}_{\pi^*}\left[\mathrm{x}_t = x, \mathrm{a}_t = a \mid \mathrm{x}_0 = x'\right],$$

i.e., $\eta(x, a)$ is the "discounted" probability that the Markov chain induced by π^* visits state x and performs action a, given the initial distribution. Assuming, without loss of generality, that π^* is stationary, we have that

$$\begin{aligned} \eta(x, a) &= \sum_{x' \in \mathcal{X}} \mathbb{P}\left[\mathrm{x}_0 = x'\right] \sum_{t=0}^{\infty} \gamma^t \mathbb{P}_{\pi^*}\left[\mathrm{x}_t = x, \mathrm{a}_t = a \mid \mathrm{x}_0 = x'\right] \\ &= \sum_{x' \in \mathcal{X}} \mathbb{P}\left[\mathrm{x}_0 = x'\right] \sum_{t=0}^{\infty} \gamma^t \mathbb{P}_{\pi^*}\left[\mathrm{a}_t = a \mid \mathrm{x}_t = x, \mathrm{x}_0 = x'\right] \mathbb{P}\left[\mathrm{x}_t = x \mid \mathrm{x}_0 = x'\right] \\ &= \pi^*(a \mid x) \sum_{a' \in \mathcal{A}} \eta(x, a'), \end{aligned}$$

where the last equality follows from noting that

$$\sum_{a\in\mathcal{A}}\eta(x,a)=\sum_{x'\in\mathcal{X}}\mathbb{P}\left[\mathrm{x}_0=x'\right]\sum_{t=0}^{\infty}\gamma^t\mathbb{P}_{\pi^*}\left[\mathrm{x}_t=x\mid\mathrm{x}_0=x'\right].$$

In summary, we showed that the linear programming formulation provides the optimal cost-to-go function, J^*, as the solution to the primal problem in (5.29), and the optimal policy, π^*, (implicitly) as the solution to the dual problem in (5.30). Our derivations also show that, as we would expect, that neither J^* nor π^* depend on the initial distribution μ considered.

5.4.4 Examples

We now revisit the examples from Section 5.2.3, illustrating the application of the different solution methods to solve the corresponding decision problems.

3-state example

In Section 5.2.3 we analyzed two policies, π and π' and concluded that, for $\gamma=0.99$, the policy π was more advantageous. We also hinted that, for the 3-state MDP, π was the optimal policy.

We now use the tools developed in this section to establish this statement. We start by verifying that the value functions computed in Section 5.2.3 indeed correspond to J^π and $J^{\pi'}$. Starting with the function J^π computed before, we have that

$$\mathsf{T}_\pi J^\pi=\boldsymbol{c}_\pi+\gamma\boldsymbol{P}_\pi\boldsymbol{J}^\pi=\begin{bmatrix}1\\0\\1\end{bmatrix}+0.99\begin{bmatrix}0&1&0\\0&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}1\\0\\100\end{bmatrix}=\begin{bmatrix}1\\0\\100\end{bmatrix}.$$

As for $J^{\pi'}$,

$$\mathsf{T}_{\pi'}J^{\pi'}=\boldsymbol{c}_{\pi'}+\gamma\boldsymbol{P}_{\pi'}\boldsymbol{J}^{\pi'}=\begin{bmatrix}\frac{1}{2}\\0\\1\end{bmatrix}+0.99\begin{bmatrix}0&0&1\\0&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}99.5\\0\\100\end{bmatrix}=\begin{bmatrix}99.5\\0\\100\end{bmatrix}.$$

We can conclude that the functions J^π and $J^{\pi'}$ are, indeed, the fixed points of T_π and $\mathsf{T}_{\pi'}$, respectively. It follows that π' is not the optimal policy, since $J^{\pi'}(0)>J^\pi(0)$. On the other hand, the Q-function for π comes

$$\boldsymbol{Q}^\pi=\boldsymbol{C}+\gamma\boldsymbol{P}\boldsymbol{J}^\pi=\begin{bmatrix}0&\frac{1}{2}\\0&0\\1&1\end{bmatrix}+0.99\begin{bmatrix}0&100\\0&0\\100&100\end{bmatrix},$$

where, given an arbitrary function $J:\mathcal{X}\to\mathbb{R}$, we write $\boldsymbol{P}\boldsymbol{J}$ to denote the matrix

$$\boldsymbol{P}\boldsymbol{J}=\begin{bmatrix}\boldsymbol{P}_{a_1}\boldsymbol{J}&\boldsymbol{P}_{a_2}\boldsymbol{J}&\dots&\boldsymbol{P}_{a_{|\mathcal{A}|}}\boldsymbol{J}\end{bmatrix}. \tag{5.31}$$

Since

$$\pi_g^Q(x)\overset{\text{def}}{=}\operatorname*{argmin}_{a\in\mathcal{A}}Q^\pi(x,a)=\pi(x),$$

we can finally conclude from Proposition 5.8 that π is the optimal policy.

Hiring a computer engineer

We revisit the hiring scenario (page 5.2.3) to illustrate the application of both value and policy iteration. As discussed in Section 5.2.3, for $N = 2$ the hiring problem can be represented as an MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$ where

- $\mathcal{X} = \big\{(B,1), (B,2), (\bar{B},2), H\big\}$;
- $\mathcal{A} = \big\{H, \bar{H}\big\}$;
- The transition probabilities are

$$\boldsymbol{P}_H = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{P}_{\bar{H}} = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

- The cost function can be represented in matrix form as

$$\boldsymbol{c} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & 0 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}.$$

Starting from the initial estimate $\boldsymbol{J}^{(0)} = \boldsymbol{0}$, we get (in matrix notation),

$$\boldsymbol{J}^{(1)} = \mathsf{T}\boldsymbol{J}^{(0)} = \min\{\boldsymbol{c} + \gamma \boldsymbol{P}\boldsymbol{J}^{(0)}\},$$

where the min is taken row-wise and $\boldsymbol{PJ}$ is defined as in (5.31). The function $J^{(1)}$ thus comes

$$\boldsymbol{J}^{(1)} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

Following Algorithm 5.2 we repeat the computation with $J^{(1)}$ to get

$$\boldsymbol{J}^{(2)} = \begin{bmatrix} 0.475 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

and $\boldsymbol{J}^{(3)} = \boldsymbol{J}^{(2)}$. We can conclude that the algorithm has converged to the function

$$\boldsymbol{J}^{*} = \begin{bmatrix} 0.475 \\ 0 \\ 1 \\ 0 \end{bmatrix},$$

which matches the cost-to-go function informally discussed in Section 5.2.3. It follows from Theorem 5.4 that the optimal policy is

$$\boldsymbol{\pi}^* = \operatorname{argmin}\{\boldsymbol{c} + \gamma \boldsymbol{P}\boldsymbol{J}^*\} = \begin{bmatrix} 0 & 1 \\ p & 1-p \\ q & 1-q \\ r & 1-r \end{bmatrix},$$

for arbitrary $p, q, r \in [0, 1]$. Any triplet (p, q, r) yields an optimal policy, meaning that there are multiple optimal policies (an infinite number of them) for this MDP.

Let us now run policy iteration in this problem. We start with an arbitrary initial policy—for example, the uniform policy

$$\boldsymbol{\pi}^{(0)} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \\ 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix},$$

and compute $\boldsymbol{J}^{(0)} = (\boldsymbol{I} - \gamma \boldsymbol{P}_{\pi^{(0)}})^{-1} \boldsymbol{c}_{\pi^{(0)}}$ to get

$$\boldsymbol{J}^{(0)} = \begin{bmatrix} 0.4875 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

The greedy policy with respect to $\boldsymbol{J}^{(0)}$ is given by

$$\boldsymbol{\pi}^{(1)} = \operatorname{argmin}\{\boldsymbol{c} + \gamma \boldsymbol{P}\boldsymbol{J}\} = \begin{bmatrix} 0 & 1 \\ p & 1-p \\ q & 1-q \\ r & 1-r \end{bmatrix}$$

where, once again, p, q and r can take any value in $[0, 1]$. Another iteration yields the cost-to-go function

$$\boldsymbol{J}^{(1)} = \begin{bmatrix} 0.475 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

and the corresponding greedy policy is $\pi^{(2)} = \pi^{(1)}$. The algorithm terminates.

Two observations are in order. First, as discussed before, the policy in states $(B, 2)$, $(\bar{B}, 2)$ and H is irrelevant, since both actions yield the same transitions and costs. Hence the existence of multiple optimal policies. More generally, let $\mathcal{M}$ be some MDP in which two actions a and b are equivalent (they yield the same transitions and costs). Then, if a is an optimal action in some state x, so is b. In fact, any policy π such that $\pi(a \mid x) = 1 - \pi(b \mid x)$ is optimal.

A second observation is that the two algorithms used—value iteration and policy iteration—exhibit similar performance in this example. In particular, they take approximately the same number of iterations and both converge to the optimal

cost-to-go function/policy. This is often not the case, as illustrated in the following examples.

$\diamond$

We conclude this example by analyzing the optimal policy for the hiring example when $N = 3$ and $N = 5$. For $N = 3$, VI concludes in 4 iterations and PI concludes in 2 iterations. The resulting optimal policy can be summarized as follows.

- If $x = (B, 1)$, $\pi^*(x) = \bar{H}$ (not hiring), meaning that the manager should keep interviewing.
- If $x = (B, 2)$, the chance that the current candidate is the best is $2/3$, so $\pi^*(x) = H$ (hiring).
- If $x = (\bar{B}, 2)$, $\pi^*(x) = \bar{H}$ (not hiring), since the present is clearly not the best candidate.
- For $x \in \{(B, 3), (\bar{B}, 3), H\}$ both actions are equivalent, so any policy is optimal.

Unlike the case where $N = 2$, the current policy does depend on the result of the interview—in particular, whether or not the second candidate is hired. To better understand the rationale behind the decision, we look at the optimal Q-function for this problem:

$$\boldsymbol{Q}^* = \begin{bmatrix} 0.67 & 0.46 \\ 0.33 & 0.63 \\ 1.00 & 0.63 \\ 0.00 & 0.00 \\ 1.00 & 1.00 \\ 0.00 & 0.00 \end{bmatrix}$$

Comparing the cost-to-go for each action in each state, several aspects become apparent. First, the cost to go associated with hiring a candidate that is not the best so far is constant and equal to 1. Second, the cost of hiring the best candidate so far decreases as more candidates are interviewed. This is natural since, as more candidates are interviewed, the probability that the best candidate so far is the best candidate overall increases. Third, the cost of not hiring increases as more candidates are interviewed, since the probability of the best candidate remaining after n interviews decreases with n.

From the observations above, we can now easily interpret the optimal policy: the manager should both refrain from hiring too early and avoid hiring too late.

Running VI and PI for $N = 5$, VI terminates after 6 iterations and PI after 3 iterations. The resulting policy is qualitatively similar to the one obtained for $N = 3$: not hiring until half the candidates are interviewed, and hire the next best candidate interviewed. Repeating the process for even higher values of N does not significantly affect the results, and the resulting policy in all these cases is essentially the same.

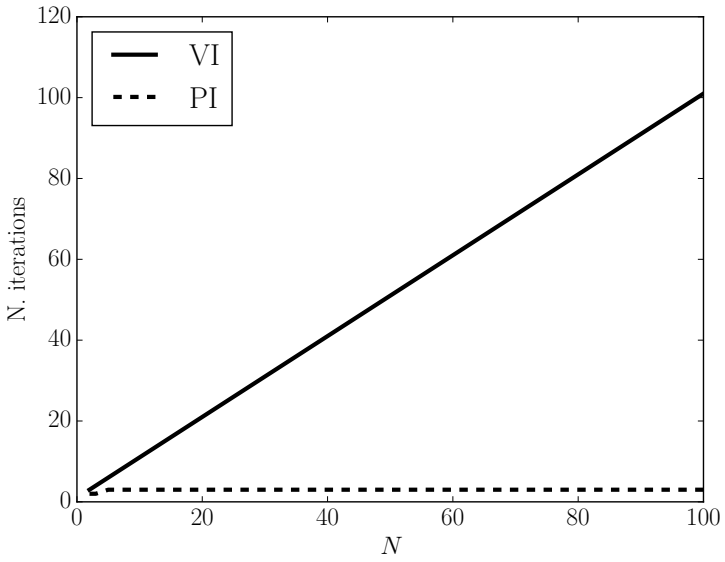


Figure 5.8 Iterations of VI and PI as N increases in the hiring example: while the number of iterations of PI remains approximately constant, the number of iterations of VI grows linearly with the value of N.

Interestingly, the number of iterations taken by each of the two algorithms shows an important difference between the two, illustrated in Fig. 5.8: PI takes approximately a constant number of iterations, while the number of iterations of VI grows linearly with N. We discuss the comparative performance of both methods in the next example.

Epidemic control

We use value and policy iteration to compute the optimal policy for the epidemic control problem. For $N = 10$, we run policy iteration with random initial policy and obtain an optimal policy after 3 iterations that can be summarized as follows.[5]

- For any $x \notin \{0, N\}$ (i.e., except in the extreme cases), $\pi^*(x) = (0.1, 0.5)$. In other words, 10% of the population should be quarantined, to avoid excessive infection; 50% should be subject to treatment, to improve healing and minimize contagion.
- In the extreme case where $x = 0$ (i.e., no one is infected), all actions $(0.1, a)$ are optimal. In other words, 10% of the population should be quarantined to prevent infection, while any level of treatment is optimal. This result can be understood by noting that (i) the cost associated to treatment is proportional

[5] The values obtained correspond to $\lambda_0 = \mu_0 = \frac{1}{N}$, $K = \frac{\lambda_0}{(N-1)\lambda_0 + \mu_0}$ and $c_Q = c_T = \frac{1}{2}c_I = 0.05$.

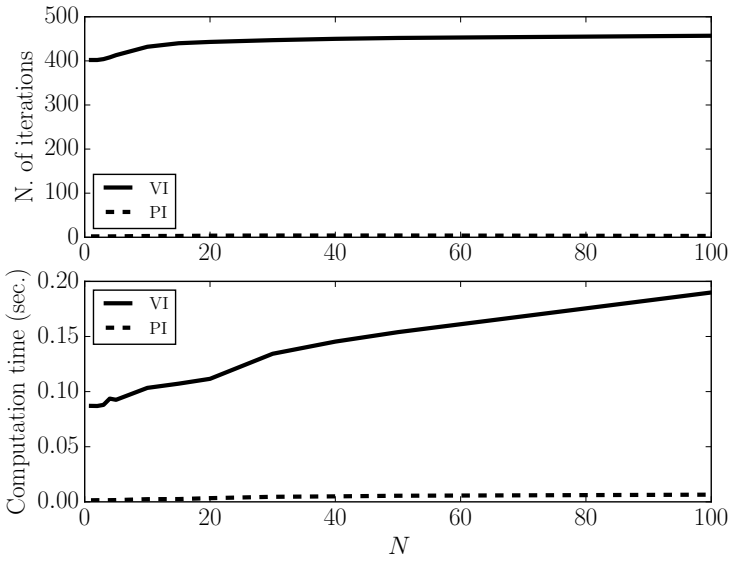


Figure 5.9 Number of iterations and computation time to convergence for both VI and PI in the epidemic control example.

to the amount of infected people, which is 0 if no one is infected; (ii) treatment impacts only healing and contagion, and has no effect if no one is infected.

- Conversely, if $x = N$ (i.e., everyone is infected), $\pi^*(x) = (0, 0.1)$. On one hand, quarantine measures only prevent infection of healthy people, and has no effect if everyone is already infected, explaining why no quarantine measures are adopted. Additionally, both the cost of treatment and and its impact on healing grow with x. In the extreme situation where $x = N$, where there is no contagion or infection, the cost is maximal and the impact moderate, so the amount of treatment should be kept small to balance cost and benefit.

Let us use the epidemic control example to compare how VI and PI perform as N grows. We run Algorithms 5.2 and 5.4 for different values of N and measured both the average computation time and the number of iterations taken by each of the two algorithms.[6] The results are depicted in Fig. 5.9 and prompt several important observations.

First of all, in this example the number of iterations of both algorithms remains approximately constant as N grows. However, there is a significant difference between the number of iterations in each of the two methods—with PI requiring two orders of magnitude less iterations to complete. This behavior is often observed

[6]The reported times were measured in a computer equipped with a 3 GHz Intel Core i7 processor and 16 GB of RAM memory.

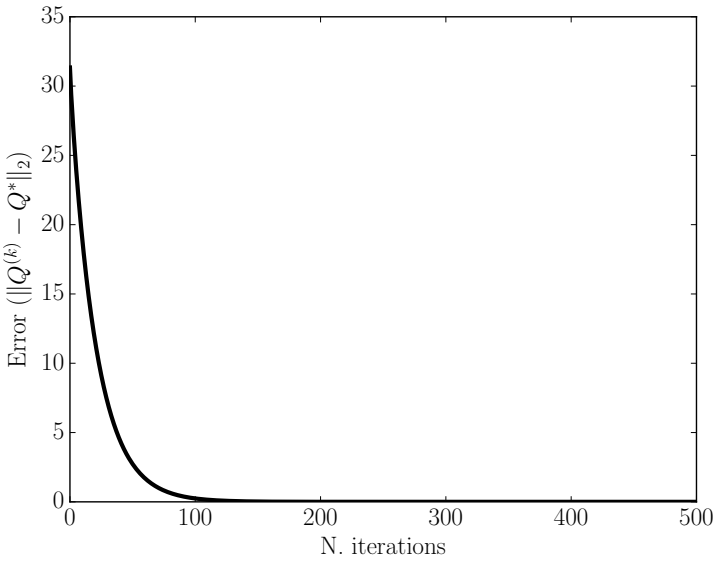


Figure 5.10 Convergence of $Q^{(k)}$ to Q^*, which is governed by the contraction factor of H, i.e., γ.

in practice: while PI can—in the worst case—require a number of iterations that is exponential in $|\mathcal{X}|$, it is often the case that it converges after a small number of iterations.

Another important observation is that the amount of time taken by the algorithms grows with N. This is also to be expected, since both cost and transition matrix grow with N and its manipulation is, therefore, more time consuming. Note, however, that the time per iteration of VI is smaller than that of PI: when $N = 100$ one iteration of PI takes around 2.2ms per iteration, while one iteration of VI takes around 0.4ms. This is to be expected: Algorithm 5.4 requires a matrix inversion, which is a time-consuming operation. Alternatively, we could use Algorithm 5.5. The advantage in time per iteration, however, comes at the cost of the number of iterations, since Algorithm 5.5 requires a larger number of iterations due to the imprecise estimation of J.

Finally, Fig. 5.10 illustrates the practical implications of the fact that H is a contraction, showing the convergence rate of $Q^{(k)}$ to Q^* as we run Algorithm 5.3.

Fault detection

Using any of the solution methods surveyed above, we can now compute the optimal policy for the fault detection system. The decision problem is relatively straightforward, since the system should visit the components either by descending order of probability of failure or by increasing order of inspection time, depending on the exact value of the costs.

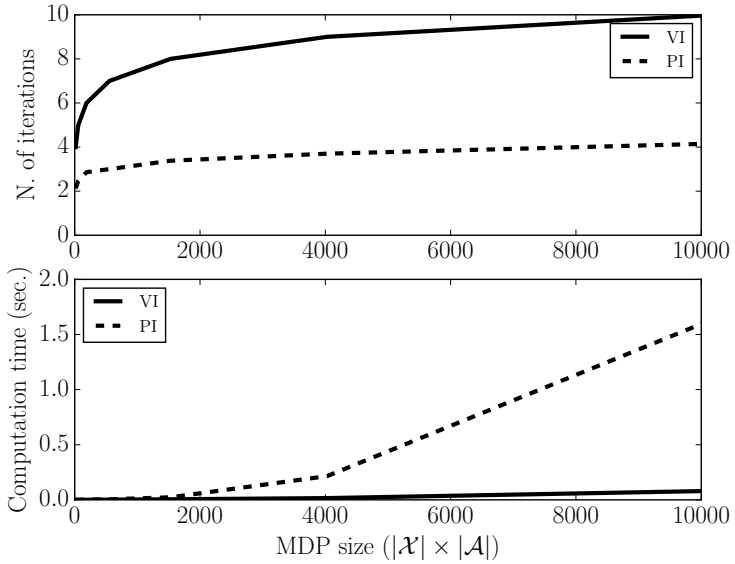


Figure 5.11 Number of iterations and computation time to convergence for both VI and PI in the fault detection example.

In our results, we set $p_n = \frac{n}{N+1}$ and $t_n = 1 + \frac{N-n+1}{N+1}$, meaning that components with higher indices have a larger probability of failure. The resulting optimal policy always inspects the uninspected component with largest index. As soon as the faulty component is identified all actions become equivalent, and the problem is essentially solved.

This domain also allows a clear observation of the computational tradeoff between PI and VI. The decision problem is quite simple, requiring few iterations of any of the two methods. On the other hand, the dimension of the problem grows exponentially with the number of components, which impacts differently the computational time taken by the two algorithms.

As in the epidemic control example, we run Algorithms 5.2 and 5.4 for different values of N and measured both the average computation time and the number of iterations taken by each of the two algorithms. The results are depicted in Fig. 5.11. The number of iterations of both algorithms grows moderately as the MDP size grows. Also, as in previous examples, PI requires less iterations to converge than VI. Interestingly, the situation differs significantly when looking at the amount of time taken by the two algorithms. As the plot clearly shows, the impact of the size of the MDP on PI is more noticeable than in VI. In fact, even requiring fewer iterations, PI ends up taking more time than VI as the MDP becomes larger.

Bus scheduling

Finally, we conclude our examples by computing the policy for the bus scheduling problem. The decision in this problem is also relatively simple: the optimal policy should wait (W) if the total number of passengers waiting at both stops is below some threshold N_0, and always dispatch the bus (S) as soon as the total number of passengers reaches or surpasses N_0. The value of N_0 depends on the particular parameters of the problem.

We illustrate the application of the linear programming approach in this domain for the case where each bus stop takes, at most, 2 passengers. In this case, the state space becomes $\mathcal{X} = \{(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)\}$.

Let μ_0 denote the uniform distribution over $\mathcal{X}$, i.e.,

$$\boldsymbol{\mu}_0 = \begin{bmatrix} \frac{1}{9} & \cdots & \frac{1}{9} \end{bmatrix}.$$

Using μ_0 as the initial distribution, we get the linear program

$$\begin{aligned} \text{maximize} \quad & \boldsymbol{\mu}_0 \boldsymbol{J} \\ \text{subject to} \quad & (\boldsymbol{I} - \gamma \boldsymbol{P}_S)\boldsymbol{J} \leq \boldsymbol{C}_{:,S} \\ & (\boldsymbol{I} - \gamma \boldsymbol{P}_W)\boldsymbol{J} \leq \boldsymbol{C}_{:,W}, \end{aligned}$$

where $\boldsymbol{C}_{:,a}$ denotes the ath column of the cost matrix $\boldsymbol{C}$. A standard simplex solver returns, after 18 iterations, the optimal cost-to-go function J^*, from which compute the following optimal policy:

- If $x \in \{(0,0),(0,1),(1,0)\}$, then the action should be W.
- Otherwise, the action should be S.

As anticipated, the optimal policy waits if the total number of passengers is below 2 and dispatches the bus otherwise.

We conclude with a brief comparison between the performance of the linear programming approach and the two dynamic programming approaches (VI and PI). Figure 5.12 depicts the running time of all three approaches in the bus problem, for different values of N. In terms of computational effort, the linear programming approach is the heaviest of the three, taking significantly longer than the two dynamic programming approaches. Moreover, as already observed in previous examples, PI takes longer than VI, although the difference between the two is less pronounced than the difference to LP.

The computational performance of LP can be understood from the fact that it uses a generic solver that fails to take advantage of the specific structure of MDPs. The DP approaches, in contrast, are build specifically to leverage the fast convergence resulting from the contraction properties of the operators T_π and H.

5.5 MDPs with large state spaces $(\star)$

The methods to solve MDPs discussed in this chapter rely on the implicit assumption that cost-to-go function or the Q-function can be represented explicitly in

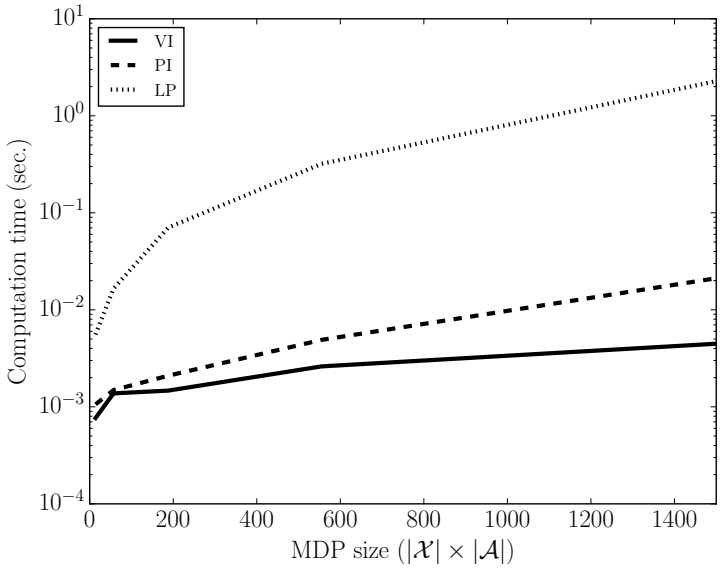


Figure 5.12 Computation time as a function of the MDP size, for the bus example.

a computer. For example, both the update rule in Algorithm 5.2 and the linear programming formulation in (5.29) explicitly manipulate $J(x)$ for every $x \in \mathcal{X}$.

However, many decision problems feature very large state and action spaces. Therefore, the associated cost-to-go function or Q-function are not amenable to explicit representation. When that is the case, solution methods must rely on some alternative compact representation that can be computationally manipulated.

$\diamond$

Suppose that we want to represent some function $F : \mathcal{X} \to \mathbb{R}$, where the domain $\mathcal{X}$ is finite but very large. Since $\mathcal{X}$ is finite, in theory we could represent F as a vector $\boldsymbol{F} \in \mathbb{R}^{|\mathcal{X}|}$. However, since $\mathcal{X}$ is very large, in practice such vector representation is computationally too expensive to manipulate. Alternatively, we consider some parameterized family $\mathcal{F} = \left\{F_{\boldsymbol{w}}, \boldsymbol{w} \in \mathbb{R}^M\right\}$, with $M \ll |\mathcal{X}|$, and instead determine the function $F_{\boldsymbol{w}^*}$ that best approximates F.

Roughly speaking, we can identify three categories of approximations:

State aggregation State aggregation breaks $\mathcal{X}$ into "macro-states" $\mathcal{X}_1, \ldots, \mathcal{X}_M$, with $\mathcal{X}_m \subset \mathcal{X}, m = 1, \ldots, M$. The sets $\mathcal{X}_m$ are nonempty and not necessarily disjoint. Each element $x \in \mathcal{X}$ is represented as a vector $\boldsymbol{\phi}(x)$, where the mth component, $\phi_m(x)$, indicates whether $x \in \mathcal{X}_m$ or not.

State aggregation approaches can be *hard*, in which case the sets $\mathcal{X}_1, \ldots, \mathcal{X}_M$ constitute a proper partition of $\mathcal{X}$ and each $x \in \mathcal{X}$ belongs to a single macro-

state $\mathcal{X}_m$, with $\phi_m(x) = \mathbb{I}[x \in \mathcal{X}_m]$; or *soft*, where each $x \in \mathcal{X}$ may belong to multiple macro-states. In soft stage aggregation, $\phi_m(x) \in [0,1], m = 1, \ldots, M$, and

$$\sum_{m=1}^{M} \phi_m(x) = 1 \tag{5.32}$$

for all $x \in \mathcal{X}$. Each value $\phi_m(x)$ can be seen as the "degree of belonging" of x to $\mathcal{X}_m$.

The resulting approximation takes the form

$$\hat{F}(x) = \sum_{m=1}^{M} \phi_m(x) w_p = \boldsymbol{\phi}^\top(x)\boldsymbol{w},$$

where each w_p is the "aggregated" value of F on $\mathcal{X}_p$.

Linear representations Linear representations describe each state $x \in \mathcal{X}$ as a vector of features, $\boldsymbol{\phi}(x)$, where each feature, $\phi_m(x), m = 1, \ldots, M$, describes some noteworthy aspect of x. These features are then combined to reconstruct the target function, F as a linear combination thereof. In other words, F is represented/approximated as a function

$$\hat{F}(x) = \sum_{m=1}^{M} \phi_m(x) w_p = \boldsymbol{\phi}^\top(x)\boldsymbol{w}.$$

Clearly, state aggregation can be seen as a special case of a linear representation, where the each feature is a function $\phi_m : \mathcal{X} \to [0,1], m = 1, \ldots, M$, and verifies (5.32).

Nonlinear representations Finally, nonlinear representations rely on more general parameterized families of functions in which the dependence of the representation on the parameter $\boldsymbol{w}$ is typically nonlinear.

Any of the three approximation approaches can be used in MDPs with large state spaces. In the continuation, we briefly discuss how the algorithms of Section 5.4 can be modified to accommodate different approximation architectures, and how the latter affect the performance of the former. We focus our discussion on value iteration algorithms (in particular, Algorithms 5.1 and 5.3) since, in a sense, they summarize most of the ideas discussed in Section 5.4.

5.5.1 State aggregation

We start by considering the prediction problem: given an MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$ and a stationary policy π, determine/approximate J^π using state aggregation.

Prediction

Let $\{\phi_m, m = 1, \ldots, M\}$ denote a collection of functions, where $\phi_m : \mathcal{X} \to [0,1], m = 1, \ldots, M$, and (5.32) holds.[7] Using state aggregation, the cost-to-go function J^π is

[7] Each function ϕ_m is associated with a macro-state $\mathcal{X}_m$ defined as $\mathcal{X}_m = \{x \in \mathcal{X} \mid \phi_m(x) > 0\}$.

approximated as a function of the form

$$J_{\boldsymbol{w}}(x) = \sum_{p=1}^{P} \phi_p(x) w_p = \boldsymbol{\phi}^\top(x)\boldsymbol{w},$$

where $\boldsymbol{w} \in \mathbb{R}^M$. Ideally, we want to compute the vector $\boldsymbol{w}^*$ such that

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^M} \|J^\pi - J_{\boldsymbol{w}}\|, \tag{5.33}$$

for some adequate norm.

For reasons that will soon become apparent, we consider the weighted norm $\|\cdot\|_\mu$, defined as follows. Let μ denote an arbitrary distribution over $\mathcal{X}$ and $F : \mathcal{X} \to \mathbb{R}$ an arbitrary function, and define

$$\|F\|_\mu^2 \stackrel{\text{def}}{=} \sum_{x\in\mathcal{X}} \mu(x) F^2(x) = \mathbb{E}_\mu\left[F^2(x)\right]. \tag{5.34}$$

The norm in (5.34) weights the contribution of each component $F(x)$ proportionally to the likelihood of the corresponding state, x, according to μ. When using the norm in (5.34), the solution to (5.33) is given by

$$\boldsymbol{w}^* = \boldsymbol{\Sigma}_\mu^{-1}\mathbb{E}_\mu\left[\boldsymbol{\phi}(x)J^\pi(x)\right], \tag{5.35}$$

where

$$\boldsymbol{\Sigma}_\mu = \mathbb{E}_\mu\left[\boldsymbol{\phi}\boldsymbol{\phi}^\top\right] = \sum_{x\in\mathcal{X}} \mu(x)\boldsymbol{\phi}(x)\boldsymbol{\phi}^\top(x).$$

The resulting function, given by

$$J(x) = \boldsymbol{\phi}^\top(x)\boldsymbol{\Sigma}_\mu^{-1}\mathbb{E}_\mu\left[\boldsymbol{\phi}(x')J^\pi(x')\right] \tag{5.36}$$

is the *orthogonal projection* of J^π on the linear span of the set $\{\phi_m, m = 1, \ldots, M\}$ (see Appendix A), which we denote as $\mathbf{Proj}_\Phi(J^\pi)$. Seen as an operator, the orthogonal projection has the following fundamental property.

Proposition 5.10. *Given an MDP $(\mathcal{X}, \mathcal{A}, \{P_a\}, c, \gamma)$, let π denote a stationary policy such that the induced chain, $(\mathcal{X}, \boldsymbol{P}_\pi)$, is* ergodic *with stationary distribution μ_π. Then, given any two functions $F_1, F_2 : \mathcal{X} \to \mathbb{R}$,*

$$\|(\mathbf{Proj}_\Phi \circ \mathsf{T}_\pi)F_1 - (\mathbf{Proj}_\Phi \circ \mathsf{T}_\pi)F_2\|_{\mu_\pi} \leq \gamma \|F_1 - F_2\|_{\mu_\pi},$$

where the projection is taken with respect to the weighted norm defined from μ_π and $(\mathbf{Proj}_\Phi \circ \mathsf{T}_\pi)$ is the composition of $\mathbf{Proj}_\Phi$ with T_π,

$$(\mathbf{Proj}_\Phi \circ \mathsf{T}_\pi)F = \mathbf{Proj}_\Phi(\mathsf{T}_\pi J).$$

Proof. See Section 5.8. □

Proposition 5.10, although seemingly complicated, states a simple but powerful result: the composite operator $(\mathbf{Proj}_\Phi \circ \mathsf{T}_\pi)$ is a contraction in the weighted norm $\|\cdot\|_{\mu_\pi}$. Recalling from Chapter 2 the meaning of the stationary distribution $\mu_\pi(x)$, we see that $\|F(x)\|_{\mu_\pi}$ weights the different components $F(x)$ proportionally to how likely it is that the chain visits x when policy π is followed.

From Proposition 5.10 we can immediate adapt Algorithm 5.1 to accommodate state aggregation, as summarized in Algorithm 5.6.

Algorithm 5.6 VI to approximate J^π using state aggregation.

Require: MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$; tolerance $\varepsilon > 0$; policy π;

1: Initialize $k = 0$, $\boldsymbol{w}^{(0)} \equiv 0$
2: **repeat**
3: $\quad \boldsymbol{w}^{(k+1)} \leftarrow \boldsymbol{\Sigma}_\mu^{-1} \mathbb{E}_{\mu_\pi} \left[\boldsymbol{\phi}(x) (\mathsf{T}_\pi J_{\boldsymbol{w}^{(k)}})(x) \right]$
4: $\quad k \leftarrow k + 1$
5: **until** $\left\| \boldsymbol{w}^{(k-1)} - \boldsymbol{w}^{(k)} \right\|_2 < \varepsilon$.
6: **return** $J_{\boldsymbol{w}^{(k)}} = \boldsymbol{\phi}^\top \boldsymbol{w}^{(k)}$

Several observations are in order. First of all, the update in Step 3 requires the ability to compute expectations w.r.t. μ_π, which may not always be easy. In practice, implementations of Algorithm 5.6 often resort to sampling. We revisit this discussion in Chapter 8.

Second, it follows immediately from our discussion in Section 5.4 that Algorithm 5.6 converges to the function

$$J_{\boldsymbol{w}^*} = (\mathbf{Proj}_\Phi \circ \mathsf{T}_\pi) J_{\boldsymbol{w}^*}$$

which, in general, is different from $\mathbf{Proj}_\Phi(J^\pi)$. However,

$$\begin{aligned} \|J_{\boldsymbol{w}^*} - J^\pi\|_{\mu_\pi} &= \|J_{\boldsymbol{w}^*} - \mathbf{Proj}_\Phi(J^\pi) + \mathbf{Proj}_\Phi(J^\pi) - J^\pi\|_{\mu_\pi} \\ &\leq \|J_{\boldsymbol{w}^*} - \mathbf{Proj}_\Phi(J^\pi)\|_{\mu_\pi} + \|\mathbf{Proj}_\Phi(J^\pi) - J^\pi\|_{\mu_\pi} \\ &\leq \gamma \|J_{\boldsymbol{w}^*} - J^\pi\|_{\mu_\pi} + \|\mathbf{Proj}_\Phi(J^\pi) - J^\pi\|_{\mu_\pi} . \end{aligned}$$

We can thus conclude that

$$\|J_{\boldsymbol{w}^*} - J^\pi\|_{\mu_\pi} \leq \frac{1}{1-\gamma} \|\mathbf{Proj}_\Phi(J^\pi) - J^\pi\|_{\mu_\pi} ,$$

which provides an error bound on how good the approximation is which depends only on the selection of the macro-states $\mathcal{X}_m, m = 1, \ldots, M$ (or, conversely, the functions $\phi_m, m = 1, \ldots, M$), and the discount γ.

Control

We now move to the problem of control: given an MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$, we want to determine/approximate Q^* using state aggregation.[8] The fundamental idea is

[8] As mentioned before, we focus on the computation of Q^*, as the process to compute J^* can be derived in a similar manner.

similar to that used to approximate J^π. However, the operator H, used to compute Q^*, is not a contraction in the weighted norm. This means that Proposition 5.10 cannot be extended directly for the computation of Q^*. Instead, we adopt an alternative to the orthogonal projection that allows a counterpart to Proposition 5.10 in the maximum norm.

Given a function $F : \mathcal{X} \to \mathbb{R}$ and an arbitrary distribution μ, define the modified projection operator $\mathbf{Proj}_\mu$ as follows

$$\mathbf{Proj}_\mu(F) = \boldsymbol{\phi}^\top(x)\mathbb{E}_\mu\left[\boldsymbol{\phi}(x')F(x')\right]. \tag{5.37}$$

Technically speaking, the operator $\mathbf{Proj}_\mu$ is not really a projection, since it is not idempotent. Nevertheless, we still refer to it as a "modified projection" to reinforce the parallel with the derivation used to approximate J^π. Note also that the operator $\mathbf{Proj}_\mu$ in (5.37) is essentially similar to that in (5.36) except for the normalizing term $\boldsymbol{\Sigma}_\mu^{-1}$.

We have the following counterpart to Proposition 5.10.

Proposition 5.11. *Given an MDP $(\mathcal{X}, \mathcal{A}, \{P_a\}, c, \gamma)$, let μ denote some distribution over $\mathcal{X}$. Then, given any two functions $F_1, F_2 : \mathcal{X} \to \mathbb{R}$,*

$$\left\|(\mathbf{Proj}_\mu \circ \mathsf{H})F_1 - (\mathbf{Proj}_\mu \circ \mathsf{H})F_2\right\|_\infty \leq \gamma \left\|F_1 - F_2\right\|_\infty.$$

Proof. See Exercise 5.6. □

We thus approximate a function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ as a matrix $\boldsymbol{W}^*$, where

$$\boldsymbol{W}^*_{:,a} = \mathbb{E}_\mu\left[\boldsymbol{\phi}(x)Q(x,a)\right]. \tag{5.38}$$

The resulting algorithm is summarized in Algorithm 5.7, a generalization of Algorithm 5.3 to accommodate state aggregation.

Algorithm 5.7 VI to approximate Q^* using state aggregation.

Require: MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$; tolerance $\varepsilon > 0$;
1: Initialize $k = 0$, $\boldsymbol{W}^{(0)} \equiv 0$
2: **repeat**
3: **for** $a \in \mathcal{A}$ **do**
4: $\boldsymbol{W}^{(k+1)}_{:,a} \leftarrow \mathbb{E}_\mu\left[\boldsymbol{\phi}(x)(\mathsf{H}Q_{\boldsymbol{W}^{(k)}})(x,a)\right]$
5: **end for**
6: $k \leftarrow k + 1$
7: **until** $\left\|\boldsymbol{w}^{(k-1)} - \boldsymbol{w}^{(k)}\right\|_\infty < \varepsilon$.
8: **return** $Q_{\boldsymbol{W}^{(k)}} = \boldsymbol{\phi}^\top \boldsymbol{W}^{(k)}$

Much like Algorithm 5.6, Algorithm 5.7 also converges to the function $Q_{\boldsymbol{W}^*}$ given by

$$Q_{\boldsymbol{W}^*}(x,a) = (\mathbf{Proj}_\mu \circ \mathsf{H})Q_{\boldsymbol{W}^*}(x,a),$$

for every $x \in \mathcal{X}$ and $a \in \mathcal{A}$. Additionally, we can use the exact same derivation to obtain the bound We can thus conclude that

$$\|Q_{\boldsymbol{W}^*} - Q^*\|_\infty \leq \frac{1}{1-\gamma} \left\|\mathbf{Proj}_\mu(Q^*) - Q^*\right\|_\infty .$$

The bound for Q^* is in the maximum norm, which is often less amenable to an easy interpretation. Additionally, as with Algorithm 5.6, the update in Step 4 is often implemented using sampling, and the performance will depend on the particular distribution μ selected. We postpone to Chapter 8 a more detailed discussion on the implementation of Algorithm 5.7 using sampling.

5.5.2 Linear and nonlinear approximations

State aggregation can be seen as a particular form of linear approximation, in which the features $\phi_m, m = 1, \ldots, M$, are restricted to lie in the interval $[0, 1]$ and add to 1. Therefore, it is not surprising that most derivations in Section 5.5.1 extend without any change to general linear approximations, in what prediction is concerned. In particular, Algorithm 5.6 and the associated theoretical results remain valid when using linear approximations, since neither depend on the restrictions on ϕ that state aggregation implies.

The same cannot be said, however, in the control setting. If we want to approximate a function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ as a function

$$\hat{Q}(x, a) = \sum_{m=1}^{M} \phi_m(x, a) w_m,$$

for some given set of *features*, $\{\phi_m, m = 1, \ldots, M\}$, we can no longer rely on Proposition 5.11, since the latter depends critically on the fact that $\phi_m(x) \in [0, 1]$ for all $x \in \mathcal{X}, m = 1, \ldots, M$. Without the guarantees of Proposition 5.11, it is even possible to observe divergence, as illustrated in the following example.

Example 5.2 Consider the MDP depicted in Fig. 5.13, corresponding to a tuple $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$ where

- $\mathcal{X} = \{1, 2, 3, 4, 5, 6\}$.
- $\mathcal{A} = \{a, b\}$.
- The transition probability matrix is given by

$$\boldsymbol{P}_a = \boldsymbol{P}_b = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0.05 & 0.95 & 0 & 0 & 0 & 0 \\ 0.05 & 0.95 & 0 & 0 & 0 & 0 \\ 0.05 & 0.95 & 0 & 0 & 0 & 0 \\ 0.05 & 0.95 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

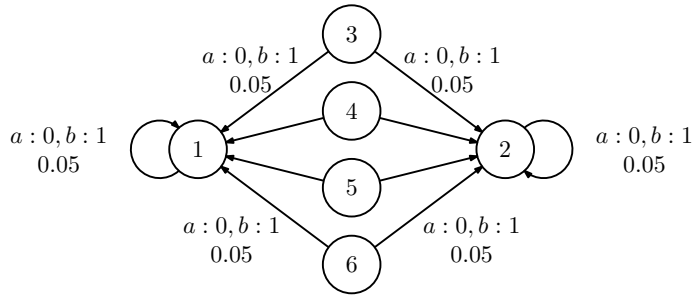


Figure 5.13 6 state MDP in which a linear approximation may diverge. All transitions from states 3, 4, 5, and 6 to state 1 occur with probability 0.05 and have a cost of 0. Similarly, all transitions from states 3, 4, 5, and 6 to state 2 occur with probability 0.95 and have a cost of 0. Some transition labels were omitted to avoid cluttering the diagram.

- The cost function can be represented as a matrix

$$\boldsymbol{C} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

- $\gamma = 0.99$.

We consider the feature set

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}, \tag{5.39}$$

where $[\boldsymbol{\Phi}]_{m,x} = \phi_m(x)$. Note that the optimal Q-function for this problem is given by $Q^*(x,a) = c(x,a)$ (since $J^* \equiv 0$). Moreover, the optimal Q-function can be represented exactly as a linear combination of the features in (5.39) by setting

$$\boldsymbol{W} = \begin{bmatrix} 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \end{bmatrix}.$$

Using Algorithm 5.7 without modification with the features in (5.39) leads to

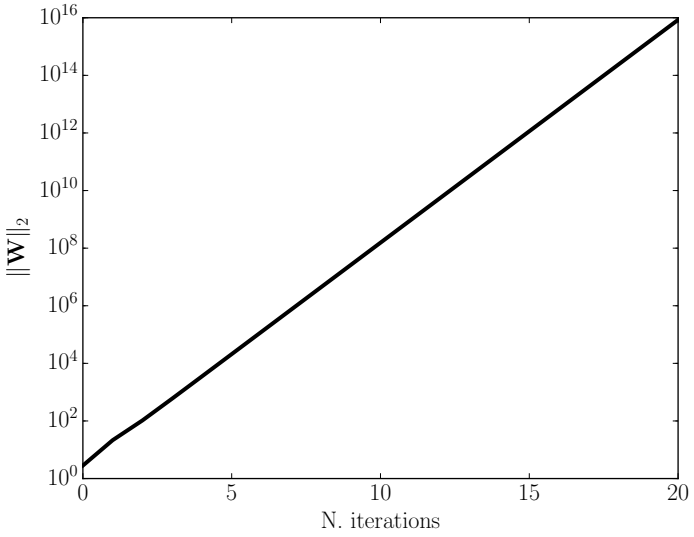


Figure 5.14 Divergence of Algorithm 5.7 with linear function approximation.

divergence, as illustrated in Figure 5.14. The plot shows how $\|\boldsymbol{W}\|$ "explodes" with the number of iterations of the algorithm (note the logarithmic scale in the y-axis).

As is to be expected, divergence like the one illustrated in Example 5.2 for a linear approximation can also be observed with nonlinear approximations.

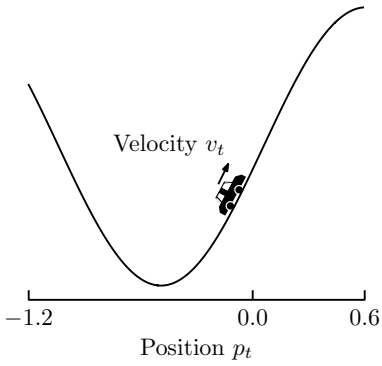
Divergence can be prevented, however, by imposing restrictions on the approximation used. Let $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ denote some target function, and let $\mathcal{F} = \left\{F_{\boldsymbol{w}}, \boldsymbol{w} \in \mathbb{R}^M\right\}$ denote some parameterized (linear or nonlinear) family of functions. We want to approximate Q by a function in $\mathcal{F}$. A mapping $\mathbf{Proj}_\infty$ is called an *averager* if

$$(\mathbf{Proj}_\infty\, Q)(x,a) = \sum_{m=1}^{M} Q(x_m, a_m)\phi_m(x,a),$$

where $\{(x_m, a_m), m = 1, \ldots, M\}$ is a predefined set of values in $\mathcal{X} \times \mathcal{A}$ and

$$\sum_{m=1}^{M} \phi_m(x,a) = 1,$$

with $\phi_m(x,a) \geq 0$ for all $(x,a) \in \mathcal{X} \times \mathcal{A}$. In other words, the representation of a target function Q in $\mathcal{F}$ is obtained by *averaging* the value of Q at some set of



$$v_{t+1} = \text{bound}[v_t + 0.001 \times \text{a}_t - g\cos(3p_t)]$$

$$p_{t+1} = \text{bound}[p_t + v_t]$$

$$v_t \in [-0.07, 0.07]$$

$$p_t \in [-1.2, 0.6]$$

Figure 5.15 The mountain car example of Boyan and Moore (1995), where an under-actuated car must go up a steep hill.

pre-specified points in $\mathcal{X} \times \mathcal{A}$.[9] We have that

$$\begin{aligned}
\|\mathbf{Proj}_\infty Q_1 - \mathbf{Proj}_\infty Q_2\|_\infty &= \max_{x,a} \left| \sum_{m=1}^{M} \phi_m(x,a)(Q_1(x_m,a_m) - Q_2(x_m,a_m)) \right| \\
&\leq \max_{x,a} \sum_{m=1}^{M} \phi_m(x,a) \left| (Q_1(x_m,a_m) - Q_2(x_m,a_m)) \right| \\
&\leq \max_{x,a} \left| (Q_1(x_m,a_m) - Q_2(x_m,a_m)) \right| = \|Q_1 - Q_2\|_\infty ,
\end{aligned}$$

and we can immediately extend the conclusion of Proposition 5.11 to the new operator $\mathbf{Proj}_\infty$. Averagers include locally weighted averaging, k-nearest neighbor approximations and, of course, state aggregation.

We conclude this section with an example that illustrates the use of approximations in a well-known continuous domain.

The mountain-car domain

Suppose that an autonomous agent must drive a car to the top of a mountain. Unfortunately, the car's engine is not powerful enough to push the car up the steep slope of the hill from rest (see Fig. 5.15), so the car must use the gravity to gain the necessary speed to reach the top of the mountain.

At each moment, the agent has available three actions: accelerate *towards* the mountain top (corresponding to the action $+1$); accelerate *away* from the mountain top (corresponding to the action -1); and do nothing (corresponding to the action 0). The dynamics of the car are summarized in Fig. 5.15, where a_t represents

[9] The original definition of averager admits the inclusion of a number of additional constants in the representation. We choose not to include those constants since they add little to the discussion and render the notation more evolved.

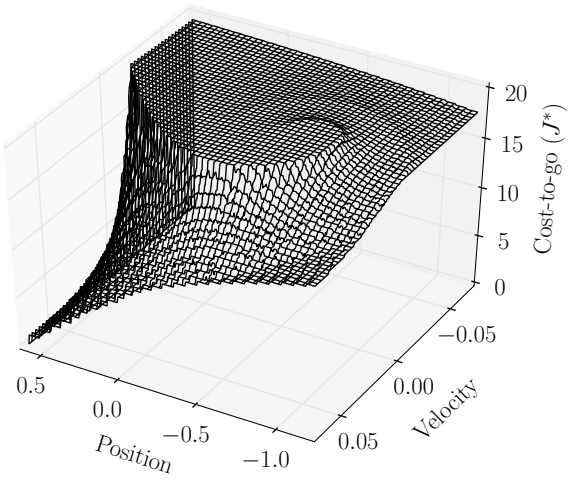


Figure 5.16 Optimal cost-to-go function for the mountain-car problem.

the action of the agent. The bound function clips the values of p_t and v_t to the corresponding intervals.[10]

The problem can be represented as a (continuous-state) MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$, where:

- $\mathcal{X} = [-1.2, 0.6] \times [-0.07, 0.07]$.
- $\mathcal{A} = \{-1, 0, 1\}$.
- The system has deterministic transitions, given by

$$v_{t+1} = \text{bound}[v_t + 0.001a_t - g\cos(3p_t)]$$
$$p_{t+1} = \text{bound}[p_t + v_t],$$

  where $g = 0.0025$ represents the gravitational acceleration.
- The cost is 1 for all states except when $p_t = 0.6$, in which the cost is 0.
- $\gamma = 0.95$.

The optimal cost-to-go function is depicted in Fig. 5.16.[11] Note that there is a discontinuity in J^* between the states in which the car does not have enough velocity to go up the hill and those in which it does.

Figure 5.17 compares the result obtained when using approximate value iteration to compute J^* with (a) 60×60 evenly spaced radial basis functions (which is

[10] Whenever the car reaches one of the ends of the scenario—i.e., whenever $p_t = -1.2$ or $p_t = 0.6$, the velocity is reset to 0.

[11] The optimal cost-to-go function was obtained by discretizing the state-space into a 300×300 grid.

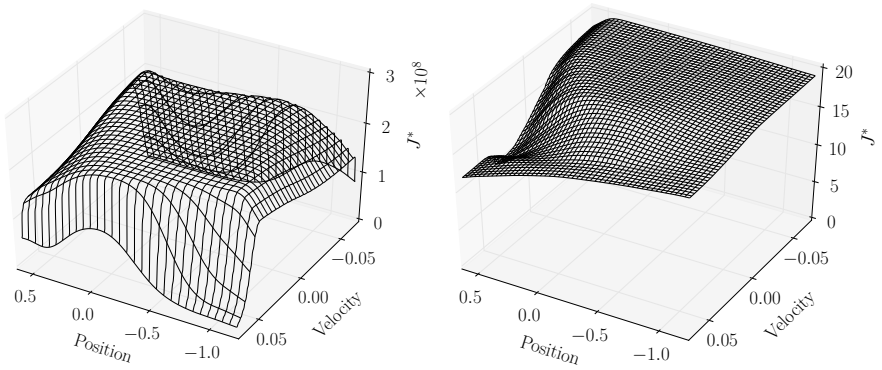


Figure 5.17 Results of approximate value iteration in the mountain-car domain. (a) Result obtained with 30×30 evenly spaced radial basis functions. The algorithm was stopped after a few iterations due to the divergence (note the scale in the z-axis). (b) Result obtained with weighted k nearest neighbors using 30×30 evenly spaced samples.

not an averager); and (b) weighted k-nearest neighbors, with 60×60 evenly spaced samples (which is an averager).

As can be observed in Fig. 5.17(a), the approach using radial basis functions diverges after a few iterations, with the values of the parameters $\boldsymbol{w} \to \infty$. The approach using an averager (Fig. 5.17(b)), on the other hand, captures the overall aspect of J^*. The resolution used (30×30 evenly spaced samples), however, does not allow for many of the details of J^* to be captured exactly.

5.5.3 A note on approximate linear programming

It is also possible to modify the linear programming approach to accommodate (linear) function approximation. In particular, given an MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$ and a linearly parameterized family $\mathcal{F} = \left\{F_{\boldsymbol{w}}, \boldsymbol{w} \in \mathbb{R}^M\right\}$, the linear program formulation becomes, in matrix notation,

$$\begin{array}{ll} \text{maximize} & \boldsymbol{\mu}\boldsymbol{\Phi}\boldsymbol{w} \\ \text{subject to} & (\boldsymbol{I} - \gamma \boldsymbol{P}_a)\boldsymbol{\Phi}\boldsymbol{w} \leq \boldsymbol{C}_{:,a}, \forall a \in \mathcal{A}, \end{array} \tag{5.40}$$

where $\boldsymbol{\Phi}$ is the matrix with m, x entry given by

$$[\boldsymbol{\Phi}]_{m,x} = \phi_m(x)$$

and $\boldsymbol{C}_{:,a}$ denotes the ath column of the cost matrix $\boldsymbol{C}$. In the linear program (5.40), the optimization variables are the components of $\boldsymbol{w}$.

In this approximate setting, the distribution μ plays a fundamental role. In a sense, μ specifies how the error $\left|J^*(x) - \boldsymbol{\phi}^\top(x)\boldsymbol{w}\right|$ is distributed across different states. Therefore, μ should be selected to reflect the relative frequency with which

different states are visited. Ideally, μ should match the stationary distribution induced by the optimal policy for the MDP. In practice, however, this is often as hard as solving the MDP itself.

Another important observation is that, since $M \ll |\mathcal{X}|$, the approximate linear programming formulation in (5.40) is significantly more compact in terms of optimization variables. However, the number of constraints remains as large as in the original problem, i.e., there is one constraint for each pair $(x, a) \in \mathcal{X} \times \mathcal{A}$. Fortunately, most of the constraints are inactive, suggesting that near-optimal solutions may be obtained by solving a reduced LP of the form

$$\begin{aligned} &\text{maximize} && \boldsymbol{\mu}\boldsymbol{\Phi}\boldsymbol{w} \\ &\text{subject to} && \boldsymbol{\eta}(\boldsymbol{I} - \gamma \boldsymbol{P}_a)\boldsymbol{\Phi}\boldsymbol{w} \leq \boldsymbol{\eta}\boldsymbol{C}_{:,a}, \forall a \in \mathcal{A}, \end{aligned}$$

where η is a sampling distribution with small support, i.e., such that $|\text{supp}(\eta)| \ll |\mathcal{X}|$. As with μ, the selection of η has a critical impact on the obtained solution. Constraints regarding states that are often visited are more critical than constraints regarding states that are seldom visited, and the principles guiding the selection of η are similar to those guiding the selection of μ.

5.6 Discussion

To conclude this section, we discuss several MDP-related topics that were briefly mentioned throughout the presentation, but whose treatment was postponed to this final section.

5.6.1 Complexity

We start by analyzing the *complexity* of Markov decision problems, i.e., how efficiently can MDP solutions be found given the *dimension* of the MDP. In Section 5.4.4 we illustrated how different methods take a different amount of time to solve an MDP, and how that time depends on the size of the MDP. However, we provided little discussion on the form of such dependence.

To analyze the complexity of Markov decision problems we make a short detour to introduce the standard nomenclature used in complexity theory. From a complexity point of view, a *decision problem* $\mathcal{D}$ is a set of instances (usually infinite), each of which having a "Yes"/"No" answer. The *size* of an instance $D \in \mathcal{D}$ is the number of bits necessary to represent it as a binary string, henceforth denoted as $|D|$. The complexity of a problem $\mathcal{D}$ is the amount of resources (time or space) required, in the worst case, to solve any instance $D \in \mathcal{D}$, as a function of $|D|$.

- A problem $\mathcal{D}$ is *polynomial* (or is in the class $\mathbf{P}$) if any instance $D \in \mathcal{D}$ can be solved in an amount of time that is polynomial in $|D|$.
- We say that a problem $\mathcal{D}$ can be *reduced* to another problem $\mathcal{D}'$ if, for any $D \in \mathcal{D}$, it is possible to determine, in an amount of time that is polynomial in $|D|$, an instance $D' \in \mathcal{D}'$ such that the answer of D' as an instance of $\mathcal{D}'$ is the same as the answer of D as an instance of $\mathcal{D}$.

- A problem $\mathcal{D}$ is *complete* for $\mathbf{P}$ (or $\mathbf{P}$-complete) if $\mathcal{D}$ is in $\mathbf{P}$ and every problem $\mathcal{D}'$ in $\mathbf{P}$ can be reduced to $\mathcal{D}$.

- Finally, a problem $\mathcal{D}$ is in class $\mathbf{NP}$ if a potential solution for an instance $D \in \mathcal{D}$ can be *verified* in an amount of time that is polynomial in $|D|$.[12]

A problem in $\mathbf{P}$ is usually accepted as "easy", meaning that a polynomial-time solution is available. In contrast, a problem in $\mathbf{NP}$ is considered "hard", as there is currently no known polynomial-time solution for it.

Given any MDP $\mathcal{M}$, deciding whether the optimal policy for $\mathcal{M}$ attains a given value K is a "Yes"/"No" decision problem that is tantamount to solving the MDP. Therefore, from a computational complexity perspective, an MDP is the set of all decision problems like the one just formulated.

Since we are able to express any finite MDP as an LP, solving an MDP cannot be harder than solving a general linear program. Linear programs are known to be $\mathbf{P}$-complete (Khachiyan, 1980), which prompts the question as to whether MDPs are also $\mathbf{P}$-complete.

The key result on MDP complexity is due to Papadimitriou and Tsitsiklis (1987). In this paper, the authors show that Markov decision problems are indeed complete for the $\mathbf{P}$ class. The result was proved by formulating the circuit value problem, known to be complete for $\mathbf{P}$, as an MDP.

A closely related question regarding MDP complexity is whether MDPs, being in $\mathbf{P}$, are *strongly polynomial*. Given an MDP $(\mathcal{X}, \mathcal{A}, \{P_a\}, c, \gamma)$, a solution method is strongly polynomial if its complexity depends only on $|\mathcal{X}|$ and $|\mathcal{A}|$ and not on the precision required to represent $\boldsymbol{P}_a$, c and γ. MDPs are strongly polynomial if they admit a strongly polynomial solution method. In an initial study into this problem, Littman (1995) investigated the complexity associated with the different solution methods—namely linear programming, value iteration and policy iteration, although no positive conclusions were established.

However, recent works have provided more definitive conclusions regarding the strong or weak polynomiality of MDP solution methods. Ye (2005) proposed the first strongly polynomial algorithm for MDPs (an interior-point method). In a subsequent work, Ye (2011) showed that both linear programming with a specific pivot rule and policy iteration are also strongly polynomial. On the other hand, Feinberg, Huang, and Scherrer (2014) showed that value iteration and modified policy iteration *are not* strongly polynomial.

5.6.2 MDP abstractions

The study of abstractions in MDPs seeks to identify redundancies in a given MDP model that can be leveraged to build an equivalent, more compact model. The latter can be more efficiently solved and its solution used to derive a solution for the former.

[12] Whether the class $\mathbf{NP}$ is distinct from the class $\mathbf{P}$ is one of the most celebrated open problems in computer science.

MDP abstractions can be roughly divided into two main families that consider, respectively, abstractions over actions and abstractions over states. We discuss these two families separately.

Temporal abstractions

Sutton, Precup, and Singh (1999) introduced the idea of *temporal abstractions* in Markov decision problems, by considering actions that extent into multiple time steps. Such temporally extended actions—also known as *options*—consist of triplet $o = (\mathcal{I}, \pi, \beta)$, where

- $\mathcal{I}$ is an *initiation set*. The initiation set contains all the states $x \in \mathcal{X}$ in which the option is available and can be initiated;
- π is a policy that describes the behavior adopted by the agent during the execution of the option;
- $\beta : \mathcal{X} \to [0, 1]$ is a *termination condition*. An option terminates (stochastically) in state $x \in \mathcal{X}$ with a probability $\beta(x)$.

A *Markov option* is an option in which the policy π is Markov.

A given set of options $\mathcal{O} = \{o_1, \dots, o_M\}$ can be incorporated in an MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$ by augmenting the action space of the agent to be $\mathcal{A}_{\text{aug}} = \mathcal{A} \cup \mathcal{O}$. Alternatively, it is also possible to define an alternative model in which $\mathcal{A}_{\text{alt}} = \mathcal{O}$. In any case, the goal of the agent is now to determine a policy that maps histories/states to distributions over $\mathcal{A}_{\text{aug}}$ or $\mathcal{A}_{\text{alt}}$ so as to minimize the total expected discounted cost.

However, taking options into account in the decision problem has two immediate implications:

- The cost associated with the execution of an option o in state x is not a function of x and o alone, but also of the amount of time elapsed until the option terminates. In particular, given an option $o = (\mathcal{I}_o, \pi_o, \beta_o)$,

$$c(x, o) = \mathbb{E}_{\pi_0}\left[\sum_{t=0}^{\mathrm{T}} \gamma^t c(\mathrm{x}_t, \mathrm{a}_t) \mid \mathrm{x}_0 = x\right],$$

  where T is the r.v. corresponding to the duration of execution of o.

- Similarly, the state in which the option terminates is also not a function of x and o alone, but also of the amount of time elapsed until the option terminates. In particular, given an option $o = (\mathcal{I}_o, \pi_o, \beta_o)$,

$$\boldsymbol{P}(y \mid x, o) = \sum_{t=1}^{\infty} \boldsymbol{P}_{\pi_o}^t(y \mid x) \mathbb{P}_\pi\left[\mathrm{T} = t\right],$$

  where $\mathbb{P}_\pi\left[\mathrm{T} = t\right]$ is the probability that the option terminates after exactly t time steps.

Strictly speaking, the resulting stochastic process, $\{(\mathrm{x}_t, \mathrm{o}_t, \mathrm{c}_t), t \in \mathbb{N}\}$ is no longer a Markov chain and requires the formalism of *semi-Markov decision problems* to be properly described and reasoned upon.

The use of temporally extended actions is particularly useful in the context of learning, as it allows the agent to gather relevant information about the decision problem more efficiently. Additionally, options typically render learning more robust and allows prior knowledge to easily be built into the agent. We revisit the use of options in learning in Chapter 8.

State-space abstraction

While temporal abstractions seek to construct "macro-actions", state abstractions instead depart from the idea of "macro-states", central in state aggregation. State abstractions identify redundancies in the state representation that can be leveraged by collapsing multiple redundant states in a single "macro-state". The resulting MDP will have a smaller state space; it can thus be solved more efficiently and the resulting policy then extended to the original MDP.

Standard approaches to state abstraction rely on the notion of *bisimulation*, originally proposed in the context of concurrent process analysis (Parks, 1981).

Bisimulation

Let $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$ denote a finite MDP, and $\overset{B}{\leftrightarrow}$ an equivalence relation induced by some partition $\{\mathcal{X}_1, \ldots, \mathcal{X}_M\}$, of $\mathcal{X}$. In other words, $x \overset{B}{\leftrightarrow} y$ if and only if x and y belong to the same subset $\mathcal{X}_m$. The relation $\overset{B}{\leftrightarrow}$ is a *bisimulation* if, for all $x, y \in \mathcal{X}$, $x \overset{B}{\leftrightarrow} y$ if and only if

- $c(x, a) = c(y, a)$ for all actions $a \in \mathcal{A}$;
- For every subset $\mathcal{X}_m$ in the partition, $\boldsymbol{P}(\mathcal{X}_m \mid x, a) = \boldsymbol{P}(\mathcal{X}_m \mid y, a)$, where
$$\boldsymbol{P}(\mathcal{X}_m \mid x, a) = \sum_{x' \in \mathcal{X}_m} \boldsymbol{P}(x' \mid x, a).$$

Two states $x, y \in \mathcal{X}$ are *bisimilar* if there is a bisimulation relation $\overset{B}{\leftrightarrow}$ such that $x \overset{B}{\leftrightarrow} y$.

The bisimilarity relation defined above corresponds to the "coarsest" of all bisimulation relations and is a powerful equivalence relation on $\mathcal{X}$. In fact, Givan, Dean, and Greig (2003) analyzed other alternative notions of equivalence—such as *action-state equivalence* or *optimal value equivalence*—and concluded they are refinements of $\sim$, in that two bisimilar states are also action-state equivalent and

optimal value equivalent. For example, if x and y are bisimilar states, then $J^*(x) = J^*(y)$.

Bisimulation can be extended to accommodate for action relabeling. An equivalence relation $\overset{B}{\leftrightarrow}$ is a *lax bisimulation* if there is an action relabeling function $l : \mathcal{X} \times \mathcal{A} \to \mathcal{A}$ such that, for all $x, y \in \mathcal{X}$, $x \overset{B}{\leftrightarrow} y$ if and only if $c(x, a) = c(y, l(x, a))$ for all actions $a \in \mathcal{A}$ and, for every subset $\mathcal{X}_m$ in the partition induced by $\overset{B}{\leftrightarrow}$,

$$\boldsymbol{P}(\mathcal{X}_m \mid x, a) = \boldsymbol{P}(\mathcal{X}_m \mid y, l(x, a)).$$

Two states $x, y \in \mathcal{X}$ are *lax bisimilar* (denoted as $x \sim y$) if there is a lax bisimulation $\overset{B}{\leftrightarrow}$ such that $x \overset{B}{\leftrightarrow} y$.

A lax bisimulation relation is a bisimulation relation on a related MDP in which the actions have been relabeled in a state-dependent manner according to the function l. For this reason, we henceforth refer only to bisimulation and bisimilar states, abusively omitting the "lax" qualifier but always keeping in mind that an action relabeling may be implicit.

The concept of lax bisimulation is closely related to the idea of MDP homomorphisms (Ravindran, 2004; Ravindran and Barto, 2002). In fact, given an MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$, two states $x, y \in \mathcal{X}$ are bisimilar if there is an MDP homomorphism from $\mathcal{M}$ to some reduced MDP $\mathcal{M}'$ in which both x and y are mapped to the same state in $\mathcal{M}'$ (J. Taylor, Precup, and Panangaden, 2008).

Unfortunately, bisimulation as defined above considers states as either bisimilar or not, and is not robust to small variations in the transition probabilities or rewards. *(Lax) bisimulation metrics* seek to address the shortcoming identified above. A bisimulation metric is a metric on $\mathcal{X}$ that seeks, on one hand, to capture the notion of bisimilarity discussed above and, on the other hand, depends smoothly on $\{\boldsymbol{P}_a, a \in \mathcal{A}\}$ and c.

$\diamond$

We start with the notion of Kantorovich distance. Given a metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and any two probability distributions μ and ν over $\mathcal{X}$, the *Kantorovich distance between μ and ν induced by d* is the value of the linear program

$$\begin{array}{ll} \text{maximize}_{\boldsymbol{m}} & (\boldsymbol{\mu} - \boldsymbol{\nu})\boldsymbol{m} \\ \text{subject to} & m_x - m_y \leq d(x, y), \text{ for all } x, y \in \mathcal{X} \\ & 0 \leq m_x \leq 1, \text{ for all } x \in \mathcal{X}, \end{array} \tag{5.41}$$

where $\boldsymbol{m} \in \mathbb{R}^{|\mathcal{X}|}$ is a column vector and $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ are the vector representations of μ and ν, respectively. We represent the Kantorovich distance between μ and ν induced by d as $K_d(\mu, \nu)$.

An interesting aspect of the Kantorovich distance is that it assigns a distance of 0 to distributions that agree on the equivalence classes induced by d.[13] Another

[13] The equivalence relation induced by a metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ corresponds to the relation $\overset{d}{\leftrightarrow}$ in which $x \overset{d}{\leftrightarrow} y$ iff $d(x, y) = 0$.

important aspect of the Kantorovich distance is that it can be effectively computed using sampling. Let $\{x_1, \ldots, x_N\}$ be a set of N independent samples from $\mathcal{X}$ distributed according to μ, and $\{y_1, \ldots, y_N\}$ a set of N independent samples from $\mathcal{X}$ distributed according to ν. Then, μ and ν can be approximated as

$$\hat{\mu}(x) = \frac{N_x}{N}, \qquad \nu(x) = \frac{M_x}{M},$$

where N_x and M_x denote the number of times that x appears in $\{x_1, \ldots, x_N\}$ and $\{y_1, \ldots, y_N\}$, respectively. Then,

$$K_d(\mu, \nu) \approx \max_{\sigma \in \text{perm}(N)} \frac{1}{N} \sum_{n=1}^{M} d(x_n, y_{\sigma(n)}), \tag{5.42}$$

where $\text{perm}(N)$ is the set of all permutations of N elements. The value in (5.42) can, in turn, be effectively computed using the Hungarian method (Kuhn, 1955).

Given a metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$—henceforth referred as the *ground metric*—we define a metric on $\mathcal{X} \times \mathcal{A}$ as

$$\delta_d((x, a), (x', a')) = \lambda_c \, |c(x, a) - c(x', a')| + \lambda_P K_d(\boldsymbol{P}(\cdot \mid x, a), \boldsymbol{P}(\cdot \mid x', a')),$$

where λ_c and λ_P are two positive constants such that $\lambda_c + \lambda_P \leq 1$ and weight the contribution of the cost and dynamics in computing the distance δ. We define the operator F that, given a metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, returns a new metric $\mathsf{F}d$ given by

$$(\mathsf{F}d)(x, x') = \max \left\{ \sup_{a \in \mathcal{A}} \inf_{a' \in \mathcal{A}} \delta_d((x, a), (x', a')), \sup_{a' \in \mathcal{A}} \inf_{a \in \mathcal{A}} \delta_d((x, a), (x', a')) \right\}.$$

The operator F has a fixed point d_{bis}. Moreover, d_{bis} is a *bisimilarity metric* for the states in $\mathcal{M}$ (Ferns, Panangaden, and Precup, 2004; J. Taylor, Precup, and Panangaden, 2008). In other words, $d_{\text{bis}}(x, y) = 0$ if and only if $x \sim y$. Finally, it can be shown that, if $\lambda_c < \gamma$,

$$|J^*(x) - J^*(y)| \leq \frac{1}{\lambda_c} d_{\text{bis}}(x, y). \tag{5.43}$$

Unfortunately, in spite of its theoretical interest, bisimilarity metrics are computationally expensive to compute. As shown by Ferns and Precup (2014), such metrics can be computed using a dynamic programming approach that is tantamount, in terms of computational effort, to solving the original problem. For this reason, the use of such metrics in model minimization remains mostly of theoretical interest.

5.6.3 Alternative optimality criteria

We briefly discuss alternative optimality criteria to the discounted cost-to-go used throughout the chapter. The bibliographical notes provide a number of references where a more detailed treatment of such criteria can be found.

Total (undiscounted) cost-to-go

One first alternative is to ignore the discount altogether, and consider the *expected total cost-to-go* (ETC) incurred by the agent:

$$TC \stackrel{\text{def}}{=} \mathbb{E}\left[\sum_{t=0}^{T} \mathrm{c}_t\right]. \tag{5.44}$$

The ETC criterion is usually considered in *finite horizon problems*, i.e., sequential decision problems involving only a finite number T of decisions. In infinite horizon problems—such as those considered so far—the value in (5.44) may not exist or go to infinity.

One common approach to deal with infinite horizon problems is to ensure that the decision problem has at least one *proper policy*. A policy is proper if it reaches an absorbing state $x \in \mathcal{X}$ with cost 0 with probability 1. Any such state is called a *terminal state*; as soon as the agent reaches a terminal state, it incurs no additional cost. In other words, proper policies are guaranteed to yield a finite total cost, and a significant part of solving (undiscounted) decision problems consists in establishing the existence of proper policies.

Average cost per-step

Another alternative criterion is the *expected average cost per-step* (EAC), defined as

$$AC \stackrel{\text{def}}{=} \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{t=0}^{\infty} \mathrm{c}_t\right]. \tag{5.45}$$

Conditions for the existence of the limit in (5.45) are well studied and can be formulated in terms of the properties of the Markov chain induced by the class of stationary policies. For example, suppose that a stationary policy π induces an ergodic chain $(\mathcal{X}, \boldsymbol{P}_\pi)$ with stationary distribution μ_π. Informally, we have

$$\lim_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{t=0}^{T} \mathrm{c}_t \mid \mathrm{x}_0 = x_0\right] \approx \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T} \mathbb{E}_{\mu_\pi}\left[\mathrm{c}_t\right] \approx \mathbb{E}_{\mu_\pi}\left[c_\pi\right].$$

The informal derivation above suggests that the average cost-per-step incurred by the agent when following a policy π is well-defined and does not depend on where the agent starts. However, formally establishing such fact requires the Markov decision problem to verify the *unichain condition*, i.e., every stationary policy must induce an irreducible recurrent chain.

The theory and practice of unichain Markov decision processes with average cost per-step is extensive and well-established. However, the unichain requirement is significantly restrictive, as many real world domains fail to verify this assumption. Moreover, Tsitsiklis (2007) showed that even *checking* the unichain condition is an NP-complete problem. Finally, the theory and practice of multichain processes is significantly less complete than that for unichain processes, making the average cost per-step a less universal criterion than the discounted cost-to-go.

5.7 Bibliographical notes

A lot can be said about Markov decision problems, going back to the works of Bellman (1954) in the 1950s. Markov decision problems have been used as models for many distinct problems involving sequential decisions in the face of uncertainty. A rather rich survey of applications can be found in the work of White (1993). The examples used in this chapter partially illustrate the wide applicability of this model, and were borrowed from several different sources. For example, the hiring example is an adaptation of the secretary example of Puterman (2005). The epidemic control example is adapted from the work of Lefèvre (1981). The fault detection example is a simplification of the problem investigated by Gluss (1959), and the bus scheduling problem is a simplification of the problem studied in the work of Ignall and Kolesar (1974).

It is impossible to cover all theory of MDPs in a single chapter, and choices had to be made regarding which material to focus on. For example, we focused on the discounted cost-to-go criterion, only brushing over other criteria in Section 5.6.3. Nevertheless, this choice was not casual. The discounted cost-to-go criteria is mathematically quite convenient, facilitating the presentation and allowing the reader to more easily grasp the main ideas behind the theory. A reader interested in exploring other criteria will find that some mathematical adjustments are necessary but, other than that, the fundamental ideas are quite similar. Moreover, the discounted cost-to-go is, probably, the most well-studied of all the criteria surveyed herein, and the one with the most complete theory.

Another choice we made was not to go into the theory of continuous-state MDPs. While there is a significant overlap in terms of results and solution methods, dealing with continuous state spaces would require significantly more involved mathematical machinery and would bring no significant gain. For our purposes, it suffices to remark that the theoretical results established in this chapter hold without significant changes in the continuous case, under certain conditions on $\{\boldsymbol{P}_a\}$ and c. For example, when $\mathcal{A}$ is finite and the support of $\boldsymbol{P}_a(\cdot \mid x)$ is countable for any $x \in \mathcal{X}$ and $a \in \mathcal{A}$, the results on this chapter hold without modification (Blackwell, 1965). the reader interested in a complete and detailed treatment of continuous-state MDPs, we refer to the book of Bertsekas and Shreve (1996).

Our discussion of approximate solutions for MDPs in Section 5.5 covers only approaches directly related to the algorithms discussed in this Chapter. The reader familiar with MDP solution methods will find particularly obvious the lack of discussion of methods based on *sampling*. The discussion of such methods is postponed to Chapters 8 and 9. Additional discussion on approximate solution methods can be found in the monograph of Mausam and Kolobov (2012).

The use of state aggregation in dynamic programming, discussed in Section 5.5.1, was pioneered in the works of Gordon (1995) and Tsitsiklis and Van Roy (1996). Example 5.2 is adapted from the work of Gordon (1995). Approximate linear programming was studied extensively by Farias and Van Roy (2003), and our presentation closely follows the discussion in Farias (2002). Although we did not go into such level of detail, Farias (2002) provides bounds on the error—both in terms of J^* and π^*—incurred by the solutions provided by approximate linear program-

ming.

We conclude by noting that there is an abundant literature on MDP theory and practice. The standard reference is, without a doubt, the book of Puterman (2005). It provides an in-depth coverage of all optimality criteria; the structure of the presentation for each criterion is quite similar to that adopted in this chapter, although with some notational differences. Other valuable references include the recent compilation of Sigaud and Buffet (2013) or, for a broader perspective on decision-making, the excellent book of Kochenderfer (2015).

5.8 Proofs

Proof of Proposition 5.1

In order to prove Proposition 5.1, we require the following intermediate result.

Lemma 5.12. *Let $\mathcal{M}$ denote some MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$ and π an arbitrary policy in Π^{HR}. Then, for any initial state $x \in \mathcal{X}$ there is a policy $\pi' \in \Pi^{\mathrm{MR}}$ (which possibly depends on x) such that*

$$\mathbb{P}_{\pi}\left[\mathrm{x}_t = y, \mathrm{a}_t = a \mid \mathrm{x}_0 = x\right] = \mathbb{P}_{\pi'}\left[\mathrm{x}_t = y, \mathrm{a}_t = a \mid \mathrm{x}_0 = x\right]. \tag{5.46}$$

for all $t \in \mathbb{N}$.

Proof. For a given initial state x, define the Markov policy

$$\pi'_t(a \mid y) = \mathbb{P}_{\pi}\left[\mathrm{a}_t = a \mid \mathrm{x}_t = y, \mathrm{x}_0 = x\right]. \tag{5.47}$$

For any $a \in \mathcal{A}$, if $x \neq y$ it trivially holds that

$$\mathbb{P}_{\pi'}\left[\mathrm{x}_0 = y, \mathrm{a}_0 = a \mid \mathrm{x}_0 = x\right] = \mathbb{P}_{\pi}\left[\mathrm{x}_0 = y, \mathrm{a}_0 = a \mid \mathrm{x}_0 = x\right] = 0.$$

On the other hand, if $x = y$,

$$\mathbb{P}_{\pi'}\left[\mathrm{x}_0 = x, \mathrm{a}_0 = a \mid \mathrm{x}_0 = x\right] = \mathbb{P}_{\pi'}\left[\mathrm{a}_0 = a \mid \mathrm{x}_0 = x\right] = \mathbb{P}_{\pi}\left[\mathrm{a}_0 = a \mid \mathrm{x}_0 = x\right].$$

We thus showed that (5.46) holds for $t = 0$. Suppose now that (5.46) holds for some t. Then, for any $a \in \mathcal{A}$ and $y \in \mathcal{X}$,

$$\begin{aligned}\mathbb{P}_{\pi'}&\left[\mathrm{x}_{t+1} = y, \mathrm{a}_{t+1} = a \mid \mathrm{x}_0 = x\right] \\ &= \mathbb{P}_{\pi'}\left[\mathrm{a}_{t+1} = a \mid \mathrm{x}_{t+1} = y, \mathrm{x}_0 = x\right] \mathbb{P}_{\pi'}\left[\mathrm{x}_{t+1} = y \mid \mathrm{x}_0 = x\right].\end{aligned} \tag{5.48}$$

The first factor on the right-hand side of (5.48) is $\pi'_{t+1}(a \mid y)$, which can be written as the probability in (5.47). The second factor, on the other hand, verifies

$$\begin{aligned}&\mathbb{P}_{\pi'}\left[\mathrm{x}_{t+1} = y \mid \mathrm{x}_0 = x\right] \\ &= \sum_{x' \in \mathcal{X}} \sum_{a' \in \mathcal{A}} \mathbb{P}_{\pi'}\left[\mathrm{x}_{t+1} = y \mid \mathrm{x}_t = x', \mathrm{a}_t = a, \mathrm{x}_0 = x\right] \mathbb{P}_{\pi'}\left[\mathrm{x}_t = x', \mathrm{a}_t = a \mid \mathrm{x}_0 = x\right] \\ &= \sum_{x' \in \mathcal{X}} \sum_{a' \in \mathcal{A}} \boldsymbol{P}(y \mid x', a') \mathbb{P}_{\pi}\left[\mathrm{x}_t = x', \mathrm{a}_t = a \mid \mathrm{x}_0 = x\right],\end{aligned}$$

where the last equality follows from the Markov property and the induction hypothesis. Putting everything together, we get that

$$\begin{aligned}
&\mathbb{P}_{\pi'}\left[\mathrm{x}_{t+1}=y,\mathrm{a}_{t+1}=a \mid \mathrm{x}_0=x\right]\\
&=\sum_{x'\in\mathcal{X}}\sum_{a'\in\mathcal{A}}\mathbb{P}_{\pi}\left[\mathrm{a}_t=a \mid \mathrm{x}_t=y,\mathrm{x}_0=x\right]\boldsymbol{P}(y \mid x',a')\mathbb{P}_{\pi}\left[\mathrm{x}_t=x',\mathrm{a}_t=a \mid \mathrm{x}_0=x\right]\\
&=\mathbb{P}_{\pi}\left[\mathrm{x}_{t+1}=y,\mathrm{a}_{t+1}=a \mid \mathrm{x}_0=x\right].
\end{aligned}$$

The conclusion follows by induction. □

According to Lemma 5.12, given a policy $\pi\in\Pi^{\mathrm{HR}}$, then for any initial state $x\in\mathcal{X}$ there is a Markov policy π' that generates the same distribution over trajectories $\{(\mathrm{x}_t,\mathrm{a}_t),t\in\mathbb{N}\}$. But then, for a fixed initial state $x\in\mathcal{X}$,

$$\begin{aligned}
J^{\pi}(x)&\stackrel{\mathrm{def}}{=}\mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t\mathrm{c}_t \mid \mathrm{x}_0=x\right]\\
&=\sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{\pi}\left[\mathrm{c}_t \mid \mathrm{x}_0=x\right]\\
&=\sum_{t=0}^{\infty}\gamma^t\sum_{y\in\mathcal{X}}\sum_{a\in\mathcal{A}}c(y,a)\mathbb{P}_{\pi}\left[\mathrm{x}_t=y,\mathrm{a}_t=a \mid \mathrm{x}_0=x\right]\\
&=\sum_{t=0}^{\infty}\gamma^t\sum_{y\in\mathcal{X}}\sum_{a\in\mathcal{A}}c(y,a)\mathbb{P}_{\pi'}\left[\mathrm{x}_t=y,\mathrm{a}_t=a \mid \mathrm{x}_0=x\right]\\
&=\sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{\pi}\left[\mathrm{c}_t \mid \mathrm{x}_0=x\right]=J^{\pi'}(x).
\end{aligned}$$

The proof is complete.

Proof of Proposition 5.2

To establish the result we note, first of all, that any real-valued function $J:\mathcal{X}\to\mathbb{R}$ can be seen as a $|\mathcal{X}|$-dimensional vector $\boldsymbol{J}$ with xth component given by $J(x)$. Then, given any two functions $J_1,J_2:\mathcal{X}\to\mathbb{R}$, we can write in matrix notation

$$\begin{aligned}
\left\|\mathsf{T}_{\pi}J_1-\mathsf{T}_{\pi}J_2\right\|_2&=\gamma\left\|\boldsymbol{P}_{\pi}\boldsymbol{J}_1-\boldsymbol{P}_{\pi}\boldsymbol{J}_2\right\|_2\\
&\leq\gamma\left\|\boldsymbol{P}_{\pi}\right\|\left\|\boldsymbol{J}_1-\boldsymbol{J}_2\right\|_2=\gamma\left\|\boldsymbol{J}_1-\boldsymbol{J}_2\right\|_2.
\end{aligned}$$

Since $\mathbb{R}^{|\mathcal{X}|}$ is a complete normed space, the Banach fixed-point theorem (see Appendix A) then guarantees that the fixed point of T_{π} exists and is unique. The conclusion follows.

Proof of Proposition 5.3

The result can be broken down in two claims: that the operator T has a unique fixed point, and that this fixed point is J^*. To show the first claim, we repeat the

steps in the proof of Proposition 5.2. Given any two functions $J_1, J_2 : \mathcal{X} \to \mathbb{R}$, we have that

$$\begin{aligned}\|\mathsf{T}J_1 - \mathsf{T}J_2\|_\infty &= \max_{x\in\mathcal{X}} \left| \min_{a\in\mathcal{A}} \left[c(x,a) + \gamma \sum_{y\in\mathcal{X}} \boldsymbol{P}_\pi(y \mid x,a) J_1(y) \right] \right. \\ &\qquad \left. - \min_{a\in\mathcal{A}} \left[c(x,a) + \gamma \sum_{y\in\mathcal{X}} \boldsymbol{P}_\pi(y \mid x,a) J_2(y) \right] \right| \\ &\leq \gamma \max_{x\in\mathcal{X}, a\in\mathcal{A}} \sum_{y\in\mathcal{X}} \boldsymbol{P}_\pi(y \mid x,a) \, |J_1(y) - J_2(y)| \\ &\leq \gamma \, \|\boldsymbol{J}_1 - \boldsymbol{J}_2\|_\infty .\end{aligned}$$

To show the second claim, we show that

(*i*) Any bounded function $J : \mathcal{X} \to \mathbb{R}$ such that $J(x) \leq (\mathsf{T}J)(x)$ for all $x \in \mathcal{X}$ verifies $J(x) \leq J^*(x)$ for all $x \in \mathcal{X}$.

(*ii*) Any bounded function $J : \mathcal{X} \to \mathbb{R}$ such that $J(x) \geq (\mathsf{T}J)(x)$ for all $x \in \mathcal{X}$ verifies $J(x) \geq J^*(x)$ for all $x \in \mathcal{X}$.

It then follows that, if $J(x) = \mathsf{T}J$, then $J = J^*$.

To show (*i*), suppose that there is a bounded function $J : \mathcal{X} \to \mathbb{R}$ such that $J(x) \leq (\mathsf{T}J)(x)$ for all x. It follows that, given any policy $\pi \in \Pi^{\mathrm{MR}}$,

$$J(x) \leq \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t \mathrm{c}_t \mid \mathrm{x}_0 = x \right] + \gamma^T \mathbb{E}_\pi \left[J(\mathrm{x}_T) \mid \mathrm{x}_0 = x \right]$$

for all $x \in \mathcal{X}$ (see Exercise 5.2). Then, for any $\varepsilon > 0$,

$$J(x) - J^\pi(x) \leq \gamma^T \mathbb{E}_\pi \left[J(\mathrm{x}_T) - J^\pi(\mathrm{x}_T) \mid \mathrm{x}_0 = x \right] \leq \varepsilon,$$

for T sufficiently large. This implies that, for any $\varepsilon > 0$,

$$J(x) \leq J^\pi(x) + \varepsilon,$$

and since ε and π are arbitrary, it must hold that

$$J(x) \leq J^*(x).$$

To show (*ii*), suppose that there is a bounded function $J : \mathcal{X} \to \mathbb{R}$ such that $J(x) \geq (\mathsf{T}J)(x)$ for all $x \in \mathcal{X}$. Equivalently, for any $\varepsilon > 0$ there is some stationary policy π such that

$$\boldsymbol{J} \geq \boldsymbol{c}_\pi + \gamma \boldsymbol{P}_\pi \boldsymbol{J} + \varepsilon \mathbf{1},$$

where the inequality is taken component-wise. But then,

$$\boldsymbol{J} \geq (\boldsymbol{I} - \gamma \boldsymbol{P}_\pi)^{-1} \boldsymbol{c}_\pi + (1-\gamma)\varepsilon (\boldsymbol{I} - \gamma \boldsymbol{P}_\pi)^{-1} \mathbf{1}$$

or, equivalently,

$$\boldsymbol{J} \geq \boldsymbol{J}^\pi + \frac{\varepsilon}{1-\gamma}\mathbf{1} \geq \boldsymbol{J}^* + \frac{\varepsilon}{1-\gamma}\mathbf{1}.$$

Since ε is arbitrary, the result follows.

Proof of Proposition 5.6

The proof uses the following auxiliary result of Puterman (2005).

Lemma 5.13. *Given a stochastic matrix $\boldsymbol{P}$ and a scalar γ such that $0 \leq \gamma < 1$, let $\boldsymbol{M}$ be a non-negative matrix with the same dimension as $\boldsymbol{P}$. Further suppose that $\boldsymbol{M} \leq \gamma \boldsymbol{P}$, where the comparison is taken component-wise. If $\boldsymbol{Q} = \boldsymbol{I} - \gamma \boldsymbol{P} + \boldsymbol{M}$, then*

$$\left\|\boldsymbol{Q}^{-1}\boldsymbol{M}\right\|_2 \leq \gamma \left\|\boldsymbol{P}\right\|_2 .$$

Proof. Since $0 \leq \boldsymbol{M} \leq \gamma \boldsymbol{P}$, it follows that $\boldsymbol{I} - \gamma \boldsymbol{P} \leq \boldsymbol{I} - \boldsymbol{M} \leq \boldsymbol{I}$ or, equivalently, $\boldsymbol{0} \leq \boldsymbol{I} - \boldsymbol{Q} \leq \gamma \boldsymbol{P}$. Therefore,

$$\left\|\boldsymbol{I} - \boldsymbol{Q}\right\|_2 \leq \gamma \left\|\boldsymbol{P}\right\|_2 < 1,$$

and the inverse $\boldsymbol{Q}^{-1}$ exists. On the other hand, since $\boldsymbol{0} \leq \boldsymbol{M} \leq \gamma \boldsymbol{P}$,

$$0 < \boldsymbol{I} + \boldsymbol{M} - \gamma \boldsymbol{P} \leq \boldsymbol{I} + \boldsymbol{M} - \gamma \boldsymbol{P} \leq \boldsymbol{I}$$

or, equivalently,

$$0 < \boldsymbol{Q} \leq \boldsymbol{I}$$

and $\left\|\boldsymbol{Q}^{-1}\right\|_2 \geq 1$. This finally yields

$$\boldsymbol{Q}^{-1}(\boldsymbol{I} - \gamma \boldsymbol{P})\boldsymbol{1} \geq (\boldsymbol{I} - \gamma \boldsymbol{P})\boldsymbol{1}.$$

Since $\boldsymbol{I} - \gamma \boldsymbol{P} = \boldsymbol{Q} - \boldsymbol{M}$, we have

$$\boldsymbol{Q}^{-1}(\boldsymbol{Q} - \boldsymbol{M})\boldsymbol{1} = \boldsymbol{1} - \boldsymbol{q}^{-1}\boldsymbol{M}\boldsymbol{1} \geq \boldsymbol{1} - \gamma \boldsymbol{P}\boldsymbol{1}$$

or, equivalently,

$$\left\|\boldsymbol{q}^{-1}\boldsymbol{M}\boldsymbol{1}\right\|_2 \leq \gamma \left\|\boldsymbol{P}\boldsymbol{1}\right\|_2 .$$

The result follows since $\boldsymbol{q}^{-1}\boldsymbol{M} \geq 0$ and thus $\left\|\boldsymbol{q}^{-1}\boldsymbol{M}\boldsymbol{1}\right\|_2 = \left\|\boldsymbol{q}^{-1}\boldsymbol{M}\right\|_2$. □

To establish the statement, we define an operator corresponding to the Gauss-Seidel update and show that operator to be a contraction. We write $\boldsymbol{P}_\pi = \boldsymbol{P}_\pi^U + \boldsymbol{P}_\pi^L$, where $\boldsymbol{P}_\pi^U$ is an upper triangular matrix and $\boldsymbol{P}_\pi^L$ is a lower triangular matrix with zeros in the main diagonal. The Gauss-Seidel update can thus be written in matrix form as

$$\boldsymbol{J}^{(k+1)} = \boldsymbol{c}_\pi + \gamma \boldsymbol{P}_\pi^U \boldsymbol{J}^{(k)} + \gamma \boldsymbol{P}_\pi^L \boldsymbol{J}^{(k+1)}$$

or, equivalently,

$$\boldsymbol{J}^{(k+1)} = (\boldsymbol{I} - \gamma \boldsymbol{P}_\pi^L)^{-1}\boldsymbol{c}_\pi + (\boldsymbol{I} - \gamma \boldsymbol{P}_\pi^L)^{-1}\gamma \boldsymbol{P}_\pi^U \boldsymbol{J}^{(k)}.$$

Given an arbitrary bounded function $J : \mathcal{X} \to \mathbb{R}$ and a stationary policy π, define the auxiliary operator T_{aux} as

$$\mathsf{T}_{\text{aux}}\boldsymbol{J} = (\boldsymbol{I} - \gamma \boldsymbol{P}_\pi^L)^{-1}\boldsymbol{c}_\pi + (\boldsymbol{I} - \gamma \boldsymbol{P}_\pi^L)^{-1}\gamma \boldsymbol{P}_\pi^U \boldsymbol{J}.$$

The operator T_{aux} has J^π as a fixed point. Moreover, for any bounded functions $J_1, J_2 : \mathcal{X} \to \mathbb{R}$,

$$\begin{aligned}\|\mathsf{T}_{\text{aux}} J_1 - \mathsf{T}_{\text{aux}} J_2\|_2 &= \left\|(\boldsymbol{I} - \gamma \boldsymbol{P}_\pi^L)^{-1} \gamma \boldsymbol{P}_\pi^U (\boldsymbol{J}_1 - \boldsymbol{J}_2)\right\|_2 \\ &\leq \left\|(\boldsymbol{I} - \gamma \boldsymbol{P}_\pi^L)^{-1} \gamma \boldsymbol{P}_\pi^U\right\|_2 \|\boldsymbol{J}_1 - \boldsymbol{J}_2\|_2 .\end{aligned}$$

The matrix $\gamma \boldsymbol{P}_\pi^U$ is non-negative and $\gamma \boldsymbol{P}_\pi^U \leq \gamma \boldsymbol{P}_\pi$. Moreover,

$$\boldsymbol{I} - \gamma \boldsymbol{P}_\pi^L = \boldsymbol{I} - \gamma \boldsymbol{P}_\pi + \gamma \boldsymbol{P}_\pi^U .$$

Lemma 5.13 finally yields

$$\|\mathsf{T}_{\text{aux}} \boldsymbol{J}_1 - \mathsf{T}_{\text{aux}} \boldsymbol{J}_2\|_2 \leq \gamma \left\|\boldsymbol{P}_\pi\right\|_2 \|\boldsymbol{J}_1 - \boldsymbol{J}_2\|_2 \leq \gamma \|\boldsymbol{J}_1 - \boldsymbol{J}_2\|_2 .$$

We showed that T_{aux} is a contraction in the 2-norm, implying that the resulting algorithm converges at a rate $\alpha = \left\|(\boldsymbol{I} - \gamma \boldsymbol{P}_\pi^L)^{-1} \gamma \boldsymbol{P}_\pi^U\right\|_2 \leq \gamma$.

Proof of Proposition 5.8

We establish the first statement of the proposition; the second statement follows directly from the definition of greedy policy and from the fact that J^* is the unique fixed point of T. We use the following result.

Lemma 5.14. *Given an MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$ and any two functions $J_1, J_2 : \mathcal{X} \to \mathbb{R}$, if $J_1(x) \leq J_2(x)$ for all $x \in \mathcal{X}$, then*

$$(\mathsf{T}_\pi J_1)(x) \leq (\mathsf{T}_\pi J_2)(x), \qquad \text{and} \qquad (\mathsf{T} J_1)(x) \leq (\mathsf{T} J_2)(x),$$

for any stationary policy π.

Proof. If $J_1(x) \leq J_2(x)$ for all $x \in \mathcal{X}$, we have, successively

$$\begin{aligned}\sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x, a) J_1(y) &\leq \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x, a) J_2(y) \\ \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x, a) J_1(y) &\leq \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x, a) J_2(y) \\ c(x, a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x, a) J_1(y) &\leq c(x, a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x, a) J_2(y).\end{aligned}$$

Taking the expectation with respect to π, we get $(\mathsf{T}_\pi J_1)(x) \leq (\mathsf{T}_\pi J_2)(x)$. Taking the maximum over a, we have that $(\mathsf{T} J_1)(x) \leq (\mathsf{T} J_2)(x)$. □

Since J^π is the fixed point of T_π,

$$J^\pi(x) = (\mathsf{T}_\pi J^\pi)(x) \geq (\mathsf{T} J^\pi)(x) = (\mathsf{T}_{\pi_g} J^\pi)(x), \tag{5.49}$$

where the last equality follows from the definition of π_g. Successively applying T_{π_g} to (5.49) and using Lemma 5.14 yields

$$J^\pi(x) \geq (\mathsf{T}_{\pi_g} J^\pi)(x) \geq (\mathsf{T}_{\pi_g} \mathsf{T}_{\pi_g} J^\pi)(x) \geq \ldots \geq J^{\pi_g}(x),$$

for all $x \in \mathcal{X}$, and the proof is complete.

Proof of Proposition 5.10

The result follows from the following Lemma, whose proof is left as an exercise.

Lemma 5.15. *Given a family of functions $\{\phi_m, m = 1, \dots, M\}$, with $\phi_m : \mathcal{X} \to \mathbb{R}, m = 1, \dots, M$, orthogonal projection operator $\mathbf{Proj}_\Phi$ is a non-expansion in the norm $\|\cdot\|_\mu$, for any distribution μ over $\mathcal{X}$, i.e.,*

$$\|\mathbf{Proj}_\Phi(F_1) - \mathbf{Proj}_\Phi(F_2)\|_\mu \leq \|F_1 - F_2\|_\mu ,$$

for any $F_1, F_2 : \mathcal{X} \to \mathbb{R}$.

Proof. See Exercise 5.5. □

On the other hand,

$$\|\mathsf{T}_\pi J_1 - \mathsf{T}_\pi J_2\|_{\mu_\pi} = \gamma \|\boldsymbol{P}_\pi \boldsymbol{J}_1 - \boldsymbol{P}_\pi \boldsymbol{J}_2\|_{\mu_\pi} \leq \gamma \|\boldsymbol{J}_1 - \boldsymbol{J}_2\|_{\mu_\pi \boldsymbol{P}_\pi} = \gamma \|\boldsymbol{J}_1 - \boldsymbol{J}_2\|_{\mu_\pi} ,$$

where the last equality follows from the definition of μ_π. Finally,

$$\|(\mathbf{Proj}_\Phi \circ \mathsf{T}_\pi)J_1 - (\mathbf{Proj}_\Phi \circ \mathsf{T}_\pi)J_2\|_{\mu_\pi} \leq \|\mathsf{T}_\pi J_1 - \mathsf{T}_\pi J_2\|_{\mu_\pi} \leq \gamma \|\boldsymbol{J}_1 - \boldsymbol{J}_2\|_{\mu_\pi} .$$

5.9 Exercises

Exercise 5.1.
Given an Markov decision process $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, x_0)$ and an arbitrary policy π, show that π induces a distribution over possible cost trajectories $\{\mathrm{c}_t, t \in \mathbb{N}\}$ in the form

$$\mathbb{P}\left[\mathrm{c}_t = c_t, t \in \mathbb{N}\right]$$
$$= \sum_{a_0 \in \mathcal{A}} p(c_0 \mid x_0, a_0)\pi(a_0 \mid \{x_0\}) \prod_{t=0}^{\infty} \sum_{x_t \in \mathcal{X}} \sum_{a_t \in \mathcal{A}} p(c_t \mid x_t, a_t)\pi(a_t \mid h_t)\boldsymbol{P}_{a_{t-1}}(x_t \mid x_{t-1}),$$

where

$$h_t = \{x_0, a_0, \dots, x_{t-1}, a_{t-1}, x_t\}$$

and

$$p(c \mid x, a) = \mathbb{I}\left[c(x, a) = c\right].$$

Exercise 5.2.
Given a function $J : \mathcal{X} \to \mathbb{R}$ such that $J(x) < (\mathsf{T}J)(x)$ for all $x \in \mathcal{X}$, show that, given any policy $\pi \in \Pi^{\mathrm{MR}}$,

$$J(x) < \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t \mathrm{c}_t \mid \mathrm{x}_0 = x\right] + \gamma^T \mathbb{E}_\pi \left[J(\mathrm{x}_T) \mid \mathrm{x}_0 = x\right],$$

for all $x \in \mathcal{X}$.

Exercise 5.3.
Prove Corollary 5.5.

Suggestion: use the Banach fixed point theorem and the fact that T_π is a contraction.

Exercise 5.4.
Given an MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$, prove that the operator H defined componentwise for any function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ as

$$(\mathsf{H}Q)(x,a) = c(x,a) + \gamma \sum_{y \in \mathcal{X}} \boldsymbol{P}(y \mid x, a) \min_{a' \in \mathcal{A}} Q(y, a'),$$

is a contraction in the maximum norm.[14]

Exercise 5.5.
Given an MDP $(\mathcal{X}, \mathcal{A}, \{\boldsymbol{P}_a\}, c, \gamma)$ and a family of functions $\{\phi_m, m = 1, \ldots, M\}$ with $\phi_m : \mathcal{X} \to \mathbb{R}, m = 1, \ldots, M$, show that

$$\|\mathbf{Proj}_\Phi(F_1) - \mathbf{Proj}_\Phi(F_2)\|_\mu \leq \|F_1 - F_2\|_\mu ,$$

for any distribution μ over $\mathcal{X}$ and any functions $F_1, F_2 : \mathcal{X} \to \mathbb{R}$, where $\mathbf{Proj}_\Phi$ is the orthogonal projection operator defined in Section 5.5.1.

Suggestion: Use the fact that the projection operator is *linear* and the fact that $\|F\|_\mu^2 = \|\mathbf{Proj}_\Phi(F)\|_\mu^2 + \|\mathbf{Proj}_{\Phi^\perp}(F)\|_\mu^2$ for any function $F : \mathcal{X} \to \mathbb{R}$.

Exercise 5.6.
Prove Proposition 5.11.

[14] The maximum norm of a vector $\boldsymbol{x} \in \mathbb{R}^N$ is given by

$$\|\boldsymbol{x}\|_\infty = \max\{|x_n|, n = 1, \ldots, N\}.$$