Introduction to Markov Decision Processes

Martin L. Puterman and Timothy C. Y. Chan

February 17, 2022

Chapter 4

Finite Horizon Models

"Begin at the beginning," the King said, very gravely, "and go on till you come to the end: then stop."
Lewis Carroll, Alice in Wonderland, 1832-1898

In this chapter, we focus on models in which the planning horizon, N, is finite and fixed, and decision makers evaluate policies on the basis of their expected total reward. By a *fixed* horizon length, we mean that the decision maker has pre-specified the number of decision epochs in the planning horizon. We distinguish this situation from models in which the planning horizon length has finite expected length but its actual length may vary from realization to realization, or possibly be unbounded. In such settings, the realized horizon length may depend on the actions of a decision maker such as in the the clinical decision making problem (Section 3.5) or the Grid World example (Section 3.7). Note it is often possible to convert a variable horizon length problem with bounded total length into a fixed horizon problem by introducing a zero-reward absorbing state. An example of this transformation is given in the case study in Section 10.3.2.

Recall that a policy π is a sequence of decision rules $(d_1, d_2, \ldots, d_{N-1})$ that determines an action choice at each decision epoch $n, n = 1, \ldots, N-1$ either deterministically or stochastically. Although policies can be quite general (recall the taxonomy in

Chapter 2), Section 2.4 showed that in the special case of a one-period problem there always exists a Markovian deterministic policy that is optimal. It turns out that this result holds in the general finite horizon setting under mild assumptions, such as when S is finite and A_s is finite for all $s \in S$. Thus, in this chapter, much of our exposition is based on Markovian deterministic policies.

The results in this chapter, while important in their own right for the analysis of finite horizon models, also motivate the analysis of infinite horizon models in later chapters. In particular, this chapter introduces the following fundamental concepts:

- Value of a policy
- Optimality equations
- Backward induction or dynamic programming
- Existence of optimal policies
- Optimality of structured policies.

4.1 The Expected Total Reward of a Policy

4.1.1 The Value of a Policy

We define the *value* of a policy $\pi = (d_1, d_2, \ldots, d_{N-1}) \in \Pi^{\text{HR}}$ in the finite horizon setting under the expected total reward criterion to be

$$v^{\pi}(s) = E^{\pi}\left[\sum_{n=1}^{N-1} r_n(X_n, Y_n, X_{n+1}) + r_N(X_N) \middle| X_1 = s\right] \tag{4.1}$$

for each $s \in S$, where X_n is the system state and Y_n is the action chosen at decision epoch n. Recall that if the policy is deterministic, then Y_n becomes $d_n(X_n)$. To compute $v^{\pi}(s)$, we introduce the concept of forward and backward recursions. Equation (4.1) holds for all classes of policies but the precise equations for computing the value differs between classes of policies under consideration. In Section 4.1.3 we provide a recursion for evaluating a Markovian deterministic policy and in Section 4.1.4 we provide a recursion for evaluating a history-dependent randomized policy.

4.1.2 Forward and Backward Recursions

Here, we compare two basic recursions underlying the analysis of finite horizon Markov decision processes. Let $v_n^{\pi}(s)$ be the expected total reward obtained using policy π, starting in state s, from decision epoch n to the end of the planning horizon. That is,

$$v_n^{\pi}(s) := E^{\pi}\left[\sum_{i=n}^{N-1} r_i(X_i, Y_i, X_{i+1}) + r_N(X_N) \middle| X_n = s\right] \tag{4.2}$$

is the expected total reward in an $N-n$ period problem that starts at state s at decision epoch n. In this notation, it follows that the expected total reward of policy π, $v^\pi(s)$, is equal to $v_1^\pi(s)$. It is straightforward to see that v_1^π can be written in terms of v_2^π:

$$\begin{aligned} v_1^\pi(s) &= E^\pi[r_1(X_1,Y_1,X_2) + \sum_{n=2}^{N-1} r_n(X_n,Y_n,X_{n+1}) + r_N(X_N)\big|X_1 = s] \\ &= E^\pi[r_1(X_1,Y_1,X_2) + v_2^\pi(X_2)\big|X_1 = s]. \end{aligned} \tag{4.3}$$

In other words, the value of policy $\pi = (d_1,\ldots,d_{N-1})$ is the expected reward obtained by choosing action Y_1, determined by decision rule d_1 in epoch 1 in state s, plus the expected total reward from epoch 2 in state X_2 to the end of the planning horizon, using decision rules $d_2,\ldots,d_{N-1}$. State X_2 is random since it is determined by the probability transitions out of state s when decision rule d_1 is applied. Note that the relationship in (4.3) is valid for any pair of successive decision epochs. Indeed, for a general $n \in \{1,\ldots,N-1\}$, if we define $v_N^\pi(X_N) := r_X(X_N)$, then we can write $v_n^\pi(s)$ as

$$\begin{aligned} v_n^\pi(s) &= E^\pi[r_n(X_n,Y_n,X_{n+1}) + \sum_{i=n+1}^{N-1} r_i(X_i,Y_i,X_{i+1}) + r_N(X_N)\big|X_n = s] \\ &= E^\pi[r_n(X_n,Y_n,X_{n+1}) + v_{n+1}^\pi(X_{n+1})\big|X_n = s]. \end{aligned} \tag{4.4}$$

Since the quantity $v_n^\pi(s)$ represents the expected total reward from decision epoch n onward, it is often referred to as the *cost-to-go* function when the rewards represent costs.

The idea of starting at v_1^π and then proceeding to v_2^π and so on using equations (4.4) is called a *forward recursion* because we think about the value at decision epoch n as depending on the value at the next decision epoch $n+1$. Forward recursions will be useful for deriving certain analytical representations, but cannot be used directly for computation because at decision epoch n, we do not yet know the value at epoch $n+1$.

In contrast to forward recursion, *backward recursion* uses the same recursive relationship (4.4) but in the reverse order. That is, we start at the end of the planning horizon and work backwards to the first decision epoch. Backward recursions will be the basis for calculations.

Writing (4.4) for $n = N-1$, we have

$$\begin{aligned} v_{N-1}^\pi(s) &= E^\pi[r_{N-1}(X_{N-1},Y_{N-1},X_N) + v_N^\pi(X_N)|X_{N-1} = s] \\ &= E^\pi[r_{N-1}(X_{N-1},Y_{N-1},X_N) + r_N(X_N)|X_{N-1} = s]. \end{aligned} \tag{4.5}$$

Since we know both expressions inside the expectation, we can evaluate $v_{N-1}^\pi(s)$. With $v_{N-1}^\pi(s)$ for all $s \in S$ readily computable, we can repeat the above process to evaluate $v_{N-2}^\pi(s)$ and so forth. Computing $v_n^\pi(\cdot)$ for $n = 1,2,\ldots,N-1$ in this way is the basis for the calculations in the next section and in many later parts of the book. Notice that this computation builds on the one-period model from Section 2.4, which presents a simple method to compute an optimal action in a given decision epoch, assuming that the next epoch provides the terminal reward. Equation (4.5) is exactly equation (2.31) when we set $N = 2$. Recall that in the one-period model, Markovian and history-dependent policies are the same. In general, we use Y_n to distinguish the history-dependent case with a probability distribution over actions versus $d_n(X_n)$ to indicate a deterministic mapping from state X_n to a specific action $d_n(X_n)$.

4.1.3 Evaluating a Markovian deterministic policy

We now show how to compute the value of a Markovian deterministic policy $\pi = (d_1, d_2, \ldots, d_{N-1})$ and illustrate the computation with an example. Since a Markovian deterministic policy generates a Markov reward process (Section 2.2.4), the calculations also apply to any Markov reward process. The idea is to break down an $(N-1)$-period problem into a sequence of $N-1$ one-period problems identical to those in Section 2.4. The computations can be summarized as follows.

Algorithm 4.1: The Finite Horizon Policy Evaluation Algorithm for Markovian Deterministic Policies

```
1 Choose $\pi = (d_1, \ldots, d_{N-1}) \in \Pi^{\mathrm{MD}}$. Set $n = N$ and $u_N(s) = r_N(s)$ for all $s \in S$.
2 while $n > 1$ do
3     $n \leftarrow n - 1$
4     for $s \in S$ do
5         Evaluate $u_n(s)$ according to
              $u_n(s) = \sum_{j \in S} p_n(j|s, d_n(s))(r_n(s, d_n(s), j) + u_{n+1}(j)).$    (4.6)
6 return $u_1(s)$ for all $s \in S$.
```

Some comments about this algorithm follow:

1. In the algorithm, we use the dummy variable u to distinguish the workings of the algorithm from the concept of the value function; this separation allows us to formalize their connection below. Indeed, Proposition 4.1 shows that this algorithm finds the expected total reward of policy π for the whole planning horizon and from decision epoch n onwards (i.e., that $u_n(s) = v_n^\pi(s)$ for all $n = 1, \ldots, N-1$ and $s \in S$).

2. Suppose there are M states in S. Each iteration of the algorithm (step 4) requires on the order of M operations (multiplications and additions) to compute $u_n(s)$ for a given s. Since there are M states and the algorithm lasts $N-1$ iterations, the algorithm requires $O(NM^2)$ operations. In terms of storage, each iteration

requires the storage of M values of $u_n(s)$, so storing all iterates results in total storage of $O(NM)$. However, if one stores only the values from the most recent iteration, then storage requirements are $O(M)$. While this book will not focus much on such algorithmic details, it is important to understand that efficient implementation can have a significant impact on computation.

3. If r_n does not depend on the subsequent state, equation (4.6) simplifies to

$$u_n(s) = r_n(s, d_n(s)) + \sum_{j \in S} p_n(j|s, d_n(s))u_{n+1}(j). \tag{4.7}$$

Policy evaluation with Algorithm 4.1

The following example illustrates the use of Algorithm 4.1 to evaluate a specified Markov deterministic policy in the two-state model of Example 2.1.

Example 4.1. Let $N = 3$ and consider the Markovian deterministic policy $\pi = (d_1, d_2)$ where

$$d_1(s_1) = a_{1,2}, \quad d_1(s_2) = a_{2,2}$$

and

$$d_2(s_1) = a_{1,1}, \quad d_2(s_2) = a_{2,1}.$$

The algorithm proceeds as follows.

1. Set $n = 3$. Since $r_3(s) = 0$ for both states s, set $u_3(s_1) = u_3(s_2) = 0$.
2. Set $n = 2$ and

$$\begin{aligned} u_2(s_1) &= p_2(s_1|s_1, a_{1,1})(r_2(s_1, a_{1,1}, s_1) + u_3(s_1)) + p_2(s_2|s_1, a_{1,1})(r_2(s_1, a_{1,1}, s_2) + u_3(s_2)) \\ &= 0.8(5 + 0) + 0.2(-5 + 0) = 3 \end{aligned}$$

and

$$\begin{aligned} u_2(s_2) &= p_2(s_1|s_2, a_{2,1})(r_2(s_2, a_{2,1}, s_1) + u_3(s_1)) + p_2(s_2|s_2, a_{2,1})(r_2(s_2, a_{2,1}, s_2) + u_3(s_2)) \\ &= 0 + 1(-5 + 0) = -5. \end{aligned}$$

3. Set $n = 1$ and

$$\begin{aligned} u_1(s_1) &= p_2(s_1|s_1, a_{1,2})(r_2(s_1, a_{1,2}, s_1) + u_2(s_1)) + p_2(s_2|s_1, a_{1,2})(r_2(s_1, a_{1,2}, s_2) + u_2(s_2)) \\ &= 0 + 1(5 + (-5)) = 0 \end{aligned}$$

and

$$\begin{aligned} u_1(s_2) &= p_2(s_1|s_2, a_{2,2})(r_2(s_2, a_{2,2}, s_1) + u_2(s_1)) + p_2(s_2|s_2, a_{2,2})(r_2(s_2, a_{2,2}, s_2) + u_2(s_2)) \\ &= 0.4(20 + 3) + 0.6(-10 + (-5)) = 0.2 \end{aligned}$$

4. Since $n = 1$, stop.

These calculations establish that $v_1^\pi(s_1) = 0$ and $v_1^\pi(s_2) = 0.2$. For comparison, let's solve this problem using *forward* recursion. Following Section 2.2.4, we must determine the distribution of states and actions under this policy. Let's consider the first case where the process starts in state s_1. In the first epoch, d_1 chooses action $a_{1,2}$, which results in a deterministic transition to state s_2. In the second epoch, at state s_2, d_2 chooses $a_{2,1}$, which results in a deterministic self-transition. Thus, the only possible realization of the process is $(s_1, a_{1,2}, s_2, a_{2,1}, s_2)$. The total reward along this sample path is 0, in agreement with Example 4.1.

If the process starts in s_2, then several realizations are possible, as summarized in Table 4.1 with the corresponding probabilities and total rewards.

Table 4.1: Realizations of states and actions, probabilities and total rewards under policy π starting in state s_2 in Example 4.1.

Realization	Probability	Total Reward
$(s_2, a_{2,2}, s_2, a_{2,1}, s_2)$	0.6	$-10 - 5 = -15$
$(s_2, a_{2,2}, s_1, a_{1,1}, s_1)$	$0.4 \times 0.8 = 0.32$	$20 + 5 = 25$
$(s_2, a_{2,2}, s_1, a_{1,1}, s_2)$	$0.4 \times 0.2 = 0.08$	$20 - 5 = 15$

It follows that the expected total reward starting in s_2 equals $0.6(-15)+0.32(25)+0.08(15) = 0.2$, also in agreement with Example 4.1. Notice that for large N, the set of possible realizations could be huge, and so Algorithm 4.1, which implements backward recursion, would be preferred to explicitly evaluating all the possible state and action realizations required in forward recursion.

Confirming that Algorithm 4.1 computes the value of a policy*

Next, we provide a laborious step-by-step proof that Algorithm 4.1 produces the value of a policy defined by equation (4.1). Our proof uses the following basic result, which is sometimes referred to as the *Law of Iterated Expectations.*

Lemma 4.1. Let X and Y be random variables. Then

$$E[X] = E_Y[E[X|Y]]$$

For a concrete example of this lemma, let X and Y be random variables that denote the lifetime earnings and years of post-secondary education, respectively, of a randomly chosen individual in the world. Computing the average earnings of all individuals in the world $E[X]$, can be done by computing the average earnings conditioned on the number of years of post-secondary education $E[X|Y]$, and then averaging these values based on the probability distribution of post-secondary education years $P(Y)$. For example, if everybody has between 0 and 10 years of post-secondary education, then $E[X] = E_Y[E[X|Y]] = \sum_{y=0}^{10} E[X|Y=y]P(Y=y)$.

Proposition 4.1. Let $\pi = (d_1, \dots, d_{n-1}) \in \Pi^{\text{MD}}$. Suppose $u_n(s)$ is computed using Algorithm 4.1, $v^\pi(s)$ is defined as in (4.1) and $v_n^\pi(s)$ is defined as in equation (4.2). Then

1. $u_n(s) = v_n^\pi(s)$ for $n = 1, \dots, N$ and all $s \in S$.
2. $v^\pi(s) = v_1^\pi(s)$ for all $s \in S$.

Proof. We prove the first part by induction. Clearly, the result holds for $n = N$ since

$$u_N(s) = r_N(s) = E^\pi[r_N(X_n)|X_n = s] = v_N^\pi(s)$$

for all $s \in S$.

Now, assume for $t = n+1, \dots, N$ and for all $s \in \mathcal{S}$,

$$u_t(s) = E^\pi\left[\sum_{i=t}^{N-1} r_i(X_i, d_i(X_i), X_{i+1}) + r_N(X_N)\middle| X_t = s\right] = v_t^\pi(s). \tag{4.8}$$

From Algorithm 4.1 and the induction hypothesis, we have

$$\begin{aligned}
u_n(s) &= \sum_{j\in S} p_n(j|s, d_n(s))[r_n(s, d_n(s), j) + u_{n+1}(j)] \\
&= \sum_{j\in S} p_n(j|s, d_n(s))[r_n(s, d_n(s), j) + v_{n+1}^\pi(j)] \\
&= E^\pi\left[r_n(X_n, d_n(X_n), X_{n+1}) + v_{n+1}^\pi(X_{n+1})|X_n = s\right] \\
&= E^\pi\left[r_n(X_n, d_n(X_n), X_{n+1}) + E^\pi\left[\sum_{i=n+1}^{N-1} r_i(X_i, d_i(X_i), X_{i+1}) + r_N(X_N)\middle| X_{n+1}\right]\middle| X_n = s\right] \\
&= E^\pi\left[r_n(X_n, d_n(X_n), X_{n+1})|X_n = s\right] + E^\pi\left[\sum_{i=n+1}^{N-1} r_i(X_i, d_i(X_i), X_{i+1}) + r_N(X_N)\middle| X_n = s\right] \\
&= E^\pi\left[\sum_{i=n}^{N-1} r_i(X_i, d_i(X_i), X_{i+1}) + r_N(X_N)\middle| X_n = s\right] \\
&= v_n^\pi(s),
\end{aligned}$$

where the fifth equality follows from Lemma 4.1 and the fact that the expectation of a sum equals the sum of expectations.

The second result simply follows from the definition of $v_1^\pi(s)$. □

4.1.4 Evaluating a History-Dependent Randomized Policy*

We don't expect that you'll ever want to evaluate a history-dependent randomized policy, but in case you do, this section will show you how. Moreover the development herein will be used in the technical appendix to establish the optimality of Markovian deterministic policies in the class of history-dependent randomized policies. Exercise 7 asks you to use it to evaluate a history-dependent randomized policy in a simple example.

Let $\pi = (d_1, d_2, \dots, d_{N-1}) \in \Pi^{\mathrm{HR}}$ be a history-dependent randomized policy. This means that at decision epoch n, d_n chooses action a according to some probability distribution given by equation (2.4). The following algorithm computes the value of a fixed history-dependent randomized policy. Note that since the number of histories grows exponentially with n, policy evaluation quickly becomes computationally intractable – it requires calculating $v_n^\pi(h^n)$ for each $h^n \in H_n$, where H_n is the set of all histories up to and including decision epoch n.

Algorithm 4.2: The Finite Horizon Policy Evaluation Algorithm for History-Dependent Randomized Policies

```
1 Choose $\pi = (d_1, \dots, d_{N-1}) \in \Pi^{\mathrm{HR}}$. Set $n = N$ and $u_N(h^N) = r_N(s^N)$ for all
    $h^N = (h^{N-1}, a^{N-1}, s^N) \in H_N$.
2 while $n > 1$ do
3     $n \leftarrow n - 1$
4     for $h^n = (h^{n-1}, a^{n-1}, s^n) \in H_n$ do
5         Evaluate $u_n(h^n)$ according to
              $$u_n(h^n) = \sum_{j \in S} \sum_{a \in A_{s^n}} w_{d_n}^n(h^n, a) p_n(j|s^n, a)(r_n(s^n, a, j) + u_{n+1}(h^{n+1})), \quad (4.9)$$
          where $h^{n+1} = (h^n, a, j)$.
6 return $u_1(h^1)$ for all $h^1 \in H_1 = S$.
```

Some comments about Algorithm 4.2 follow:

1. We leave it as an exercise to show that $u_n(h^n) = v_n^\pi(h^n)$ for all $h^n \in H_n$ and $n = 1, \dots, N$.

2. In the first iteration (when $n = N$), the algorithm assigns the same value to all histories
$$h^N = (s^1, a^1, s^2, \dots, s^{N-1}, a^{N-1}, s^N)$$
that agree in state s^N. That is, if the state at decision epoch N is s, then all histories that end at s are given the same value.

3. In (4.9), recall that $w_{d_n}^n(h^n, a)$ represents the probability that action a is chosen in epoch n, given the history h^n and decision rule d_n. Since decision rules are history-dependent, the randomization distribution $w_{d_n}^n(h^n, a)$ may be different

for different histories. However, rewards and transition probabilities depend only on s^n and not the entire history. Moreover, if action a is chosen in state h^n at decision epoch n and the next transition is to state j, the history at decision epoch $n+1$ is (h^n, a, j) (see (2.3)), which is the argument of $u_{n+1}(\cdot)$.

4.2 Optimal policies and their values

A decision maker's goal is to find an optimal policy, that is, one that maximizes the expected total reward among all possible policies. In this section, we formalize the definition of optimal policies for finite horizon Markov decision processes and provide a recursion to compute them.

Definition 4.1. An *optimal policy* $\pi^* \in \Pi^{\mathrm{HR}}$ satisfies

$$v^{\pi^*}(s) \geq v^{\pi}(s) \tag{4.10}$$

for all $\pi \in \Pi^{\mathrm{HR}}$ and $s \in S$.

Definition 4.2. The *optimal value function* $v^*(s)$ is defined by

$$v^*(s) := \sup_{\pi \in \Pi^{\mathrm{HR}}} v^{\pi}(s). \tag{4.11}$$

for all $s \in S$. Similarly, we define the optimal value from epoch n to the end of the planning horizon as

$$v_n^*(s) := \sup_{\pi \in \Pi^{\mathrm{HR}}} v_n^{\pi}(s). \tag{4.12}$$

As a consequence of (4.10) and (4.11), when an optimal policy π^* exists, its value satisfies

$$v^{\pi^*}(s) = v^*(s) \tag{4.13}$$

for all $s \in S$. Exercise 4 provides a simple example (with a countable action set) in which an optimal policy need not exist.

In cases where there is no optimal policy, focus shifts to finding a policy that is suboptimal by an amount no greater than ε.

Definition 4.3. For any $\varepsilon > 0$, an *ε-optimal policy* $\pi^{\varepsilon} \in \Pi^{\mathrm{HR}}$ satisfies

$$v^{\pi^{\varepsilon}}(s) \geq v^*(s) - \varepsilon \tag{4.14}$$

for all $s \in S$.

Of course $v^*(s) \geq v^{\pi^\varepsilon}(s)$ for all $s \in S$. Exercise 5 asks you to establish that for any Markov decision process, there always exists an ε-optimal policy.

Finally, under mild assumptions such as having finite action sets, we can show that Markovian deterministic policies are optimal within the more general class of history-dependent randomized policies. This means that the search for an optimal policy can be restricted to searching within the class of Markovian deterministic policies.

Theorem 4.1. Suppose A_s is finite for each $s \in S$. Then there exists $\pi^* \in \Pi^{\text{MD}}$ that achieves the optimal value $v^*(s)$ for all $s \in S$. That is,

$$v^*(s) = \max_{\pi \in \Pi^{\text{MD}}} v^\pi(s). \tag{4.15}$$

Moreover, the same policy π^* achieves the optimal value starting at any epoch $n = 1, \ldots, N$

$$v_n^*(s) = \max_{\pi \in \Pi^{\text{MD}}} v_n^\pi(s) \tag{4.16}$$

Notice that not only does Theorem 4.1 speak to an optimal policy and optimal value starting from the first epoch, it establishes that the same policy is optimal from any decision epoch n to the end of the planning horizon. Richard Bellman coined the expression *The Principle of Optimality*, to describe this property of optimal policies. He wrote:

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

For a concrete interpretation, consider driving from location A to Z along the shortest path, which involves visiting locations B, C, D, ..., Y on the way. Then it must be the case that the shortest path from B to Z visits C, D, ..., Y as well in the same order. The reason is that if there is a different path from B to Z that is shorter, then we could use that when driving from A to Z, which contradicts the fact that the initial path from A to Z was shortest.

It is critical to recognize that Theorem 4.1 allows us to focus on Markovian deterministic policies in our exposition, without loss of generality. This is a fundamental result within the theory of Markov decision processes. We will provide a constructive proof below.

4.2.1 Computing Optimal Values and Finding Optimal Policies

In this section, we provide a recursion to compute $v^*(s)$ and show how to use it to find an optimal policy, π^*, assuming one exists. Given Theorem 4.1, the exposition below assumes policies are Markovian and deterministic.

In the spirit of the one-period model of Section 2.4, set $v^*_N(s) = r_N(s)$ and define $v^*_{N-1}(s)$ for all $s \in S$ by

$$v^*_{N-1}(s) = \max_{a \in A_s} \left\{ \sum_{j \in S} p_{N-1}(j|s,a)(r_{N-1}(s,a,j) + v^*_N(j)) \right\}. \tag{4.17}$$

The quantity $v^*_{N-1}(s)$ represents the maximum expected total reward achievable starting in state s in a one-period model with immediate reward $r_{N-1}(s,a,\cdot)$ and terminal reward $v^*_N(\cdot) = r^*_N(\cdot)$.

Define $A^*_{N-1,s}$ to be the set of actions at decision epoch $N-1$ that achieve the maximum in equation (4.17):

$$A^*_{N-1,s} = \arg\max_{a \in A_s} \left\{ \sum_{j \in S} p_{N-1}(j|s,a)(r_{N-1}(s,a,j) + v^*_N(j)) \right\}. \tag{4.18}$$

Now construct a decision rule $d^*_{N-1}(s)$ by setting it equal to some $a^*_{N-1} \in A^*_{N-1,s}$ for each $s \in S$. Although the maximum need not be unique, selecting any element of $A^*_{N-1,s}$ suffices. By construction, $v^{d^*_{N-1}}(s) = v^*_{N-1}(s)$, so it is an optimal decision rule at decision epoch $N-1$ in a one-period problem.

Next, define $v^*_{N-2}(s)$ for all $s \in S$ by

$$v^*_{N-2}(s) = \max_{a \in A_s} \left\{ \sum_{j \in S} p_{N-2}(j|s,a)(r_{N-2}(s,a,j) + v^*_{N-1}(j)) \right\}. \tag{4.19}$$

By the same argument as above, $v^*_{N-2}(s)$ is the maximum expected total reward achievable starting in state s in a one-period model with immediate reward $r_{N-2}(s,a,\cdot)$ and terminal reward $v^*_{N-1}(\cdot)$. But, considering the interpretation of $v^*_{N-1}(s)$, we can equivalently interpret $v^*_{N-2}(s)$ as the maximum expected total reward in a two-period problem with immediate rewards $r_{N-2}(s,a,\cdot)$ and $r_{N-1}(s,a,\cdot)$, and terminal reward $r_N(\cdot)$. Repeating this argument leads to a recursive algorithm for computing v^*.

In general, define

$$v^*_n(s) = \max_{a \in A_s} \left\{ \sum_{j \in S} p_n(j|s,a)(r_n(s,a,j) + v^*_{n+1}(j)) \right\}. \tag{4.20}$$

Let $A^*_{n,s}$ be the set of optimal actions at epoch n, those that achieve the maximum in equation (4.20):

$$A^*_{n,s} = \arg\max_{a \in A_s} \left\{ \sum_{j \in S} p_n(j|s,a)(r_n(s,a,j) + v^*_{n+1}(j)) \right\}. \tag{4.21}$$

Letting $d_n^*(s) = a_n^* \in A_{n,s}^*$ results in an optimal decision rule at epoch n. By choosing $d_n^*(s)$ to be an element of $A_{n,s}^*$ for $n = 1, 2, \ldots, N-1$ and $s \in S$, then $\pi^* = (d_1^*, d_2^*, \ldots, d_{N-1}^*)$ is an optimal policy. In other words, $v^{\pi^*}(s) = v^*(s)$.

The following algorithm formalizes the computation of an optimal value function and optimal policy.

Algorithm 4.3: The Finite Horizon Policy Optimization Algorithm

```
1 Set $n = N$ and $u_N(s) = r_N(s)$ for all $s \in S$.
2 while $n > 1$ do
3     $n \leftarrow n - 1$
4     for $s \in S$ do
5         Evaluate $u_n(s)$ according to
              $$u_n(s) = \max_{a \in A_s} \left\{ \sum_{j \in S} p_n(j|s,a)(r_n(s,a,j) + u_{n+1}(j)) \right\}. \qquad (4.22)$$
6         Set
              $$A_{n,s}^* = \arg\max_{a \in A_s} \left\{ \sum_{j \in S} p_n(j|s,a)(r_n(s,a,j) + u_{n+1}(j)) \right\}. \qquad (4.23)$$
7         Select $d_n(s) \in A_{n,s}^*$.
8 return $u_1(s)$ for all $s \in S$ and $\pi = (d_1, \ldots, d_{N-1})$.
```

Some important comments about Algorithm 4.3 follow:

1. This algorithm is often referred to as *Backwards Induction* or *Dynamic Programming.*

2. Expression (4.22) is the fundamental relationship underlying finite horizon Markov decision processes. It is often called the *Bellman Equation*, after Richard Bellman, who is regarded as the founder of dynamic programming. Others refer to it simply as the *optimality equation.*

3. Theorem 4.2 below formally states that this algorithm computes $v_n^*(s)$, the optimal expected total reward over decision epochs $n, n+1, \ldots, N-1$ starting in state s at decision epoch n, and $v^*(s)$, the optimal value function, as defined by (4.11) for all $s \in S$.

4. Any policy that chooses an action in the set $A_{n,s}^*$ for each s and n is optimal. If the sets contain a single element for all s and n, there is a unique optimal policy. Note that when there are multiple optimal policies, one might be preferred to another because it has a more interpretable structure or the value has smaller

variance (e.g., over many simulation replicates). Section 4.4 provides a systematic approach to identifying structured optimal policies.

5. Note that finding the set of optimal actions $A^*_{n,s}$ requires no additional computation. It is obtained as a consequence of evaluating the maximum in (4.22) over each action a and storing those actions that achieve the maximum.

6. As in the case of the Finite Horizon Policy Evaluation Algorithm, the Finite Horizon Policy Optimization Algorithm decomposes the multi-period optimization problem into a sequence of one-period optimization problems. This approach avoids enumerating all Markovian deterministic policies and their corresponding sample paths. If there are M states and K actions in each, the algorithm requires $(N-1)KM^2$ multiplications. When only a single optimal policy is required, one needs only store the policy elements obtained at each iteration and $v^*_n(s)$.

7. Theorem 4.1 is applicable for both finite and countable state space. Accordingly, Algorithm 4.3 is applicable to both cases. However, one cannot naively calculate $u_n(s)$ for every s when $\mathcal{S}$ is countably infinite. Practical options are to truncate the state space at a large state value (and modify the transition probabilities accordingly) or to establish some structure of the value function or optimal policy that makes it easy to identify an optimal action for all states above a certain threshold. We use the queuing service rate control application to illustrate both options in Sections 4.3.1 and 4.4.2. Another option is to approximate $v^*_n(s)$ by a parametric function function of s. See Chapter 10.

The following theorem states that Algorithm 4.3 is guaranteed to find an optimal policy that is Markovian and deterministic, along with the corresponding optimal value. The lengthy but insightful development needed to formally prove this result appears in Appendix 4.1. Combining Theorem 4.2 with Theorem 4.1 implies that Algorithm 4.3 finds an optimal policy within the most general class of history-dependent randomized policies.

Theorem 4.2. Let A_s be finite for all $s \in S$. Suppose $u_n(s)$ is computed using Algorithm 4.3, $v^*(s)$ is defined as in (4.15), and $v^*_n(s)$ is defined as in (4.16).

1. For $n = 1, 2, \dots, N$ and $s \in S$, $u_n(s) = v^*_n(s)$. In particular, $u_1(s) = v^*(s)$.

2. Suppose for $n = 1, 2, \dots, N$ and $s \in S$ that

$$\sum_{j \in S} p_n(j|s, d_n(s))(r_n(s, d_n(s), j) + u_{n+1}(j))$$
$$= \max_{a \in A_s} \left\{ \sum_{j \in S} p_n(j|s, a)(r_n(s, a, j) + u_{n+1}(j)) \right\}. \quad (4.24)$$

Then $\pi^* = (d_1, d_2, \ldots, d_{N-1})$ is an optimal policy.

Returning to the discussion surrounding the Principle of Optimality, immediately following Theorem 4.1, it follows that Algorithm 4.3 provides a method to construct optimal policies and determine optimal values from any decision epoch n onward.

When do optimal policies exist?

The astute reader will recognize that the reason we assume finite action sets above is to ensure that the maximum is attained in equation (4.22). Another condition that ensures attainment of the maximum is that A_s is a compact set for all $s \in S$ and $r_n(s, a, j)$ and $p_n(j|s, a)$ are *continuous* functions of a for all $s \in S$ and $j \in S$. However, optimal policies need not exist when A_s is countable or an open set.

We delve into this issue more formally by recognizing that the attainment of the max in equation (4.22) is sufficient for the existence of an optimal policy, regardless of whether the state space is finite, countable or continuous.

Corollary 4.1. Suppose the maximum is attained in (4.22) for $n = 1, 2, \ldots, N-1$ and $v_N^*(s) = r_N(s)$ for all $s \in S$. Then there exists an optimal policy $\pi^* = (d_1^*, d_2^*, \ldots, d_{N-1}^*) \in \Pi^{\text{MD}}$ for any $d_n^*(s) \in A_{n,s}^*$ as defined by (4.23) for $n = 1, 2, \ldots, N-1$.

4.2.2 A numerical example illustrating computation of an optimal policy

We now show how to use The Finite Horizon Policy Optimization Algorithm to find an optimal policy and the corresponding optimal value using the two-state example from Example 2.1.

Example 4.2. Let $N = 3$.
1. Set $n = 3$. Since $r_3(s) = 0$ for $s = s_1$ and $s = s_2$, set $u_3(s_1) = u_3(s_2) = 0$.
2. Set $n = 2$ and

$$u_2(s_1) = \max\{0.8(5) + 0.2(-5) + 0,\ 5 + 0\} = 5$$

and

$$u_2(s_2) = \max\{-5 + 0,\ 0.4(20) + 0.6(-10) + 0\} = 2$$

Consequently $A_{2,s_1}^* = \{a_{1,2}\}$ and $A_{2,s_2}^* = \{a_{2,2}\}$.

3. Set $n = 1$ and

$$u_1(s_1) = \max\{5 + 2,\ 0.8(5 + 5) + 0.2(-5 + 2)\} = 7.4$$

and

$$u_1(s_2) = \max\{-5+2,\ 0.4(20+5)+0.6(-10+2)\} = 5.2$$

Consequently $A^*_{1,s_1} = \{a_{1,2}\}$ and $A^*_{1,s_2} = \{a_{2,2}\}$.
4. Since $n = 1$, stop.

These calculations show that $v^*(s_1) = 7.4$ and $v^*(s_2) = 5.2$, and the unique optimal policy is $\pi^* = (d^*_1, d^*_2)$ where $d^*_1(s_1) = d^*_2(s_1) = a_{1,2}$ and $d^*_1(s_2) = d^*_2(s_2) = a_{2,2}$. This policy is unique because the optimal action set contains a single element at each decision epoch and state.

4.3 Applications

In this section, we apply the Finite Horizon Policy Optimization Algorithm to numerically find optimal policies for several of the applications in Chapter 3.

4.3.1 Queuing service rate control

Consider a finite horizon variant of the service rate control problem described at the end of Section 3.3.1. Let the horizon length be $N = 5$. We assume there are three service probabilities $a_1 = 0.2$, $a_2 = 0.4$ and $a_3 = 0.6$. Let the arrival probability be $b = 0.1$. Let the delay cost be $f(s) = s$, the cost per period of serving at rate a to be $m(a) = 10a$ and the terminal reward be $r_N(s) = 0$ for all s.

To facilitate computation, we assume the state is truncated at some large occupancy level W at which arrivals are blocked and do not enter the system. To determine an appropriate value for W, we can appeal to steady state results for an $M/M/1$ queue, since the model is a discrete-time version of an $M/M/1$ queue given a fixed choice of the service probability a. In particular, it is well-known that for service probability a and arrival probability b ($b < a$) the expected queue length and variance in queue length in an $M/M/1$ queue equals $b/(a-b)$ and $ba/(a-b)^2$, respectively. Thus, setting

$$W = \frac{b}{a_1 - b} + 4\frac{\sqrt{ba_1}}{a_1 - b}, \tag{4.25}$$

which is four standard deviations above the mean, ensures that W is reached with low probability even under the lowest service probability. Using the parameter values above and equation (4.25), we set $W = 6$.

As a result of truncating the state space at W, we add transition probabilities $p(W-1|W, a_k) = a_k$ and $p(W|W, a_k) = 1 - a_k$ for $k = 1, 2, 3$. The transition probabilities up to state $W-1$ are the same as before (see equations (3.1) and (3.2)): For $s = 1, 2, \ldots, 5$

and $k = 1, 2, 3$,

$$p(j|s, a_k) = \begin{cases} a_k & j = s - 1 \\ 0.1 & j = s + 1 \\ 1 - a_k - 0.1 & j = s. \end{cases} \tag{4.26}$$

For $s = 0$ and $k = 1, 2, 3$,

$$p(j|0, a_k) = \begin{cases} 0.1 & j = 1, \\ 0.9 & j = 0. \end{cases} \tag{4.27}$$

Given these parameter values, we can implement Algorithm 4.3. Since the rewards in this problem represent costs, we use the negative of the reward defined in Section 3.3.1, $m(a) + f(s)$, and replace the maximization with a minimization.

At termination, we obtain the optimal value function

$$v^* = (8.5, 11.5, 15.4, 19.4, 23.4, 27.4, 30.9)$$

corresponding to states $s = 0, \dots, 6$. The optimal policy consists of decision rules where for each state and epoch, the optimal action is a_1. To understand this result, notice that the cost of choosing a_3 is 6, which is also equal to the delay cost of having 6 users in the system. Thus, it is intuitive that it is optimal to choose the least costly action in every state and epoch.

However, we notice different behavior as we change the parameters. For example, consider quadratic delay costs ($f(s) = s^2$). Re-running the algorithm, we find that the optimal decision rule at epoch 1 is $d_1^* = (a_1, a_1, a_1, a_3, a_3, a_3, a_3)$. That is, if the system starts out in a state that is not too busy ($s = 0, 1, 2$), then it is optimal to use the lowest service probability. But if the system is more heavily loaded ($s = 3, \dots, 6$), then it is optimal to use the highest service probability. We observe similar behavior at epoch 2, where the optimal decision rule is $d_2^* = (a_1, a_1, a_1, a_1, a_3, a_3, a_3)$. The only difference with epoch 1 is that in state 3, the optimal decision switches from a_3 to a_1. The reasoning is that being one epoch closer to the end of the horizon, there is less likelihood that using a lower service probability will lead to a more costly system state by the end of the horizon. So, the reduction in cost due to using a lower service probability was more than the expected increase in cost due to future delays. If we consider cubic delay costs ($f(s) = s^3$), we find that the optimal action is to use the highest service probability for even less busy states. For example, $d_1^* = d_2^* = (a_1, a_1, a_3, a_3, a_3, a_3, a_3)$. Figure 4.1 illustrates these policies.

Next, we modify the cost of serving at probability a to make it quadratic ($m(a) = 10a^2$) or cubic ($m(a) = 10a^3$). Assume the delay costs are quadratic ($f(s) = s^2$). Re-running the algorithm, we find that the optimal decision rule at epoch 1 is now $d_1^* = (a_1, a_1, a_2, a_3, a_3, a_3, a_3)$. The difference from before is that at state 2, the optimal action is now the intermediate service probability. Because the cost of an action is now proportional to the square of its value, and because the action values are less than 1, the relative cost of $a_2 = 0.4$ to $a_1 = 0.2$ has decreased. So it has become more cost effective to use a higher service probability in state 2. At epoch 2, the

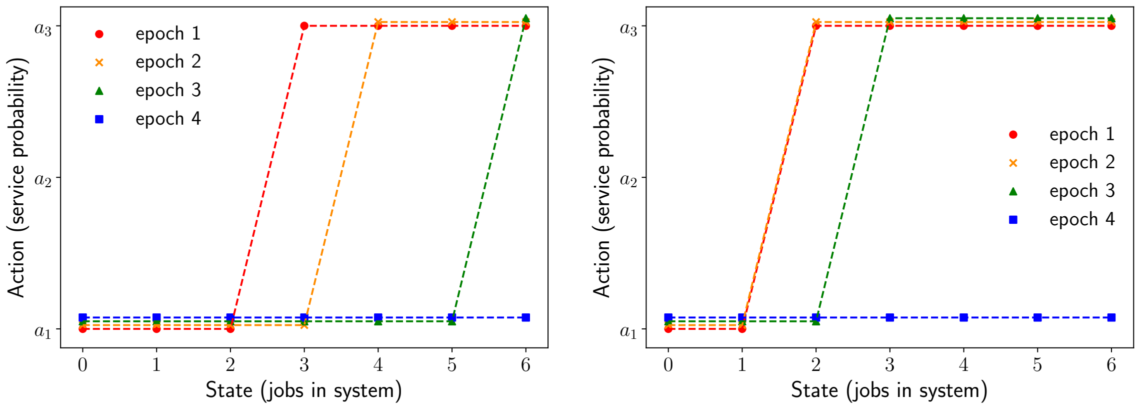

Figure 4.1: Optimal policy for linear service cost $m(a) = 10a$ and quadratic delay cost $f(s) = s^2$ (left) or cubic delay cost $f(s) = s^3$ (right).

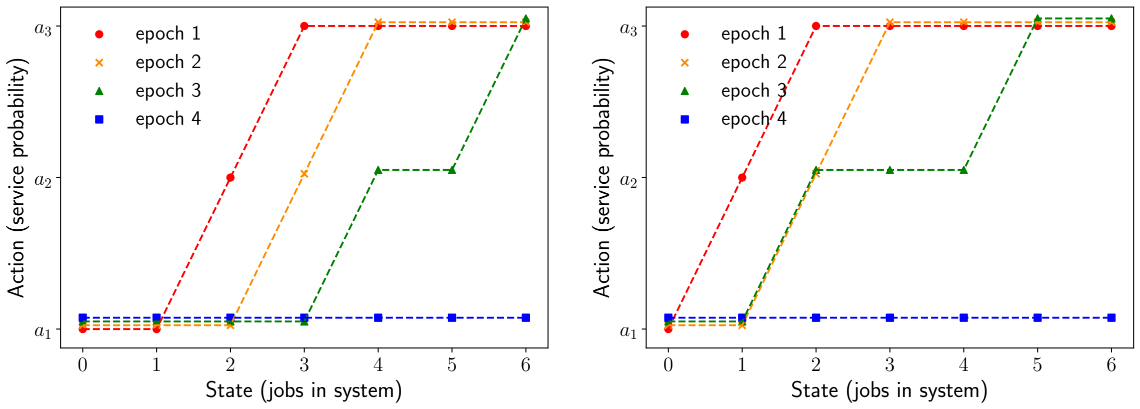

Figure 4.2: Optimal policy for quadratic delay cost $f(s) = s^2$ and quadratic service cost $m(a) = 10a^2$ (left) or cubic service cost $m(a) = 10a^3$ (right).

decision rule becomes $d_2^* = (a_1, a_1, a_1, a_2, a_3, a_3, a_3)$. Compared to d_1^*, equal or lower service probabilities are used in each state, since we are one period closer to the end of the horizon. Finally, if we use a cubic serving cost, then it becomes more cost effective to use higher service probabilities in less busy states. For example, in this case $d_1^* = (a_1, a_2, a_3, a_3, a_3, a_3, a_3)$. Note that the decision rules presented so far are *monotonic*, meaning that as the state increases the value of the completion probability is non-decreasing. Figure 4.2 illustrates these policies.

A peculiar structure arises when the delay costs are linear ($f(s) = s$) and the serving costs remain cubic ($m(a) = 10a^3$). The optimal decision rule at epoch 1 is $d_1^* = (a_1, a_1, a_2, a_2, a_2, a_2, a_1)$. This decision rule is not monotonic. We see a similar result when the serving costs are changed to $m(a) = 5a^3$ and an extra epoch is added, namely that $d_1^* = (a_1, a_2, a_2, a_3, a_3, a_3, a_2)$. Figure 4.3 illustrates these policies. In these

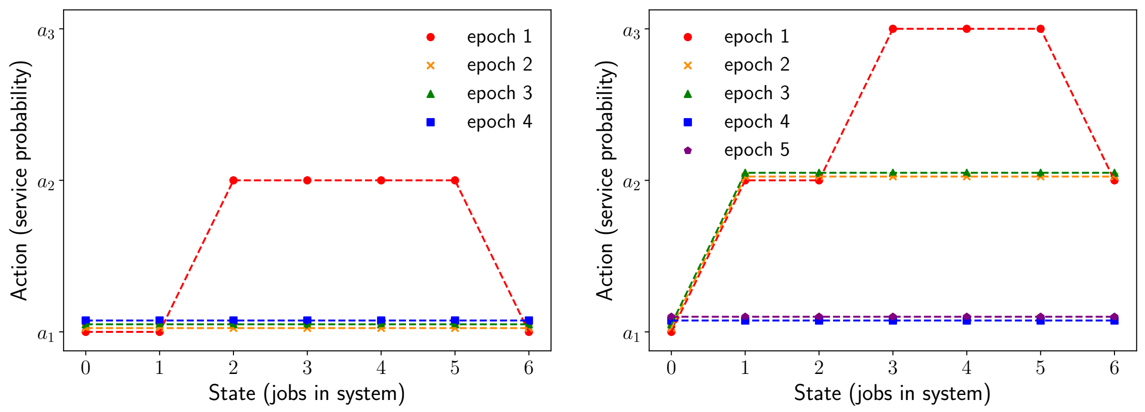

Figure 4.3: Optimal policy for linear delay cost $f(s) = s$ and cubic service costs $m(a) = 10a^3$ with $N = 5$ (left) or cubic service costs $m(a) = 5a^3$ with $N = 6$ (right). Unlike the previous example, here the policy is non-monotone at epoch 1.

cases and others, the optimal decision at state W is a slower service rate than at state $W - 1$. This result persists if we increase W. At state W, the linear delay cost is not sufficiently high to warrant the use of the highest serving probability. Plus, since the system is at capacity, there is no chance an arrival will come and increase the system's delay costs further. However, at $W - 1$, this is not the case, as delay costs can still increase with another arrival. When the delay costs were quadratic or cubic, as in previous examples, being in state W was much more undesirable, so the optimal action was a_3 in early epochs. Indeed, with serving costs $m(a) = 5a^3$, the difference in value between a_2 and a_3 is about 0.1% to 0.2% for states 2 to W. Thus, small changes to the cost structure are likely to lead to different optimal actions. Note that the lack of monotonicity only occurs in epoch 1 in Figure 4.3. However, depending on the cost structure, the optimal decision rule can be non-monotonic for later epochs (see Exercise 17).

The above example illustrates that the optimal decision rule is monotonic except at state W. It turns out this is due to the state space truncation. If we do not truncate the state space, we can prove that optimal decision rules, under certain conditions, are guaranteed to be monotonic. In the case of a truncated state space, we can guarantee monotonicity by setting the action in state W to always be the one corresponding to the highest serving probability.

4.3.2 Revenue management

In this example, we consider the revenue management problem from Section 3.1. Assume the selling season is six months and the retailer has 15 units of inventory to sell. We assume there are seven candidate prices over the course of the season, with the lowest equaling the scrap value. The non-scrap candidate prices are \$20, \$23, \$25, \$27,

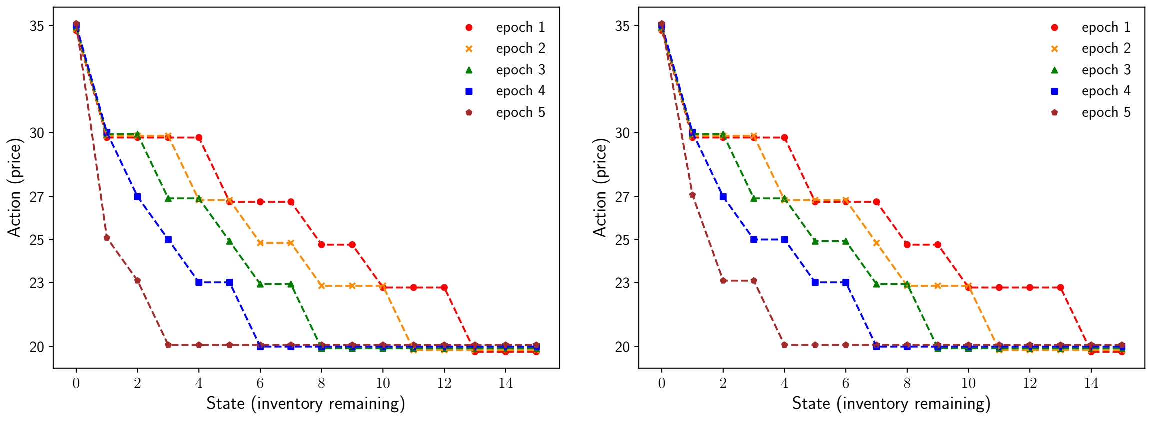

Figure 4.4: Policy across time for zero scrap value (left) and \$5 scrap value (right).

\$30 and \$35. We consider two possibilities for the scrap value, \$0 and \$5, to investigate how the solution changes as a function of the scrap value. The holding cost per item is \$2 and is calculated based on the end of month inventory. Finally, the monthly demand is Poisson distributed with rate $\lambda_{n,a} = (1.1 - 0.1n)(9 - 0.25a)$, which is linearly decreasing in both a and n.

Figure 4.4 depicts optimal policies for the two different scrap values. First, we notice that the policies are monotonic, which means that as the inventory grows, the retailer will set a lower price. For a fixed inventory level, the price also drops as we approach the end of the horizon. However, with a higher scrap value of \$5, the optimal decision rule may be willing to set higher prices at a given epoch compared to the case with zero scrap value. For example, compare the price at epoch 5 with three units of inventory remaining. When the scrap value is \$0, the optimal decision is to reduce the price to the lowest value of \$20. But when the scrap value is \$5, the optimal decision is to keep the price at \$23.

4.4 Interpretability: Optimality of Structured Policies

Interpretability refers to the ability of a user to clearly understand why a model prescribes a specific action in a specific state. This concept has become especially important as decision problems have become increasingly complex and often result in black box solutions that users may not be confident in implementing.

Along these lines, a significant stream of research that dates back to the origins of operations research in the 1950s focused on interpretability by examining the structure of optimal policies in Markov decision process applications. One of the most well-known examples comes from inventory control. Consider an inventory model with fixed ordering costs and linear holding and shortage costs. It was shown that there exists an optimal policy that takes the form of an (s, S) policy: when the inventory level falls below s, and order is placed to raise the inventory level up to S. Such policies are also referred to *min-max* policies and implemented in many commercial inventory systems. Policies of this form are easy to understand and implement. The primary challenge is to specify optimal values for the quantities s and S.

In addition to understanding the theoretical basis for the structure of an optimal policy, there is also a computational advantage to identifying the structure of an optimal policy. Knowledge that such a policy exists reduces the search for an optimal policy to a smaller class. For example, if one can compute the cost of an (s, S) policy by other analytical methods, then optimizing over s and S will provide an optimal policy.

Another example comes from linear programming. Assuming a linear programming problem has an optimal solution and its feasible region contains at least one vertex, the search for an optimal solution can be restricted to a search among the vertices. This result is closely related to the optimality of Markovian deterministic policies among the space of all history-dependent randomized policies. We make this connection between solutions to linear programs and Markov decision processes explicit in Section 5.14.

Another major class of structured optimal policies are control limit policies, which are prevalent in many applications including queuing control, maintenance, and clinical decision making.

Definition 4.4. Assume the states $s \in S$ can be ordered. A *control limit policy* is a Markovian deterministic policy with decision rules $d_n(s), n = 1, \ldots, N-1$ of the form

$$d_n(s) = \begin{cases} a_1 & s \leq \bar{s}_n \\ a_2 & s > \bar{s}_n, \end{cases}$$

where a_1 and a_2 are two distinct actions and $\bar{s}_n$ is the *control limit*. Note that the control limit can vary over decision epochs.

A control limit policy partitions the state space into two parts and chooses a different action in each part. The partition has a particularly appealing structure in that states less than or equal to $\bar{s}_n$ are mapped to one action while those larger than $\bar{s}_n$ are mapped to another action.

We begin by providing some general advice on how to demonstrate that an optimal policy is structured. Then, we provide two examples of a structured optimal policy using applications from the previous chapter.

4.4.1 A General Approach

The following series of inductive steps enables one to demonstrate the optimality of a structured policy:

1. Show that $v_N^*(s)$ has the specific form that ensures that an optimal decision rule has the desired structure.

2. Assume that for $t = N-1, N-2, \dots, n+1$, $v_t^*(s)$ has the specific form.
3. Prove that $v_n^*(s)$ has the specific form.

First, we must establish that if the optimal value function at epoch N has a specific form such as convexity or monotonicity, then there exists an optimal decision rule that has the desired structure. Then, the key is to show that the optimal value function retains this form at epoch n, given that it holds at epoch $n+1$, i.e., when $v_n^*(s)$ is derived from the Bellman equation:

$$v_n^*(s) = \max_{a \in A_s} \left\{ \sum_{p \in S} p_n(j|s,a)(r_n(s,a,j) + v_{n+1}^*(j)) \right\}$$

Mathematical induction establishes that the above procedure produces a structured optimal policy for all n. Usually, step 3 requires considerable ingenuity, as the following examples show.

4.4.2 Monotonicity of Optimal Service Rate Control Policies in Queuing Systems*

In Section 4.3.1, we observed numerically that in some parameter regimes (e.g., quadratic delay costs and linear serving costs) the optimal policy chose either the lowest service probability a_1 or highest service probability a_3, depending on the state and epoch. Furthermore, the optimal policy exhibited a “threshold” structure. That is, given a particular epoch, there was a particular state s' such that a_1 is used in all states $0 \le s \le s'$ and a_3 is used in all states $s' < s \le W$. Such a policy is an example of a control limit policy. In other parameter regimes (e.g., quadratic delay costs and cubic serving costs), the optimal policy included all three actions, with larger values of the action being optimal in larger state values. We also noted that in some cases (e.g., linear delay costs and cubic serving costs), this monotonicity did not hold at the largest state W.

In this section, we formalize these numerical observations theoretically. We show that under certain reasonable conditions, an optimal policy for the service rate control problem is a monotone policy. That is, assuming ordered states and actions, we show that the optimal action a_s^* is non-decreasing in s. To avoid the non-monotonicity due to state space truncation, we focus on the situation where the state space is countably infinite.

Definition 4.5. A decision rule d is *monotone* if $d(s)$ is non-decreasing or non-increasing in s. A policy $\pi = (d_1, d_2, \dots, d_N)$ is monotone if it consists of monotone decision rules $d_1, d_2, \dots, d_N$.

First, we write the Bellman equation (4.22) for this model. Like in the previous subsection, we transform the problem from reward maximization to cost minimization. So, for $n = 1, \dots, N-1$ and $s = 1, 2, \dots$ we have

$$v_n(s) = \min_{a \in A_s} \{f(s) + m(a) + av_{n+1}(s-1) + (1-a-b)v_{n+1}(s) + bv_{n+1}(s+1)\}. \quad (4.28)$$

Let $A^*_{n,s}$ define the set of minimizers of (4.28):

$$A^*_{n,s} = \arg\min_{a \in A} \{f(s) + m(a) + av_{n+1}(s-1) + (1-a-b)v_{n+1}(s) + bv_{n+1}(s+1)\}. \quad (4.29)$$

Since this set may have multiple elements, we define $a^*_{n,s} = \min\{a \mid a \in A^*_{n,s}\}$ as the smallest element in the set.

Due to the slight differences in transition probabilities at the boundary state $s = 0$, the Bellman equation becomes

$$\begin{aligned} v_n(0) &= \min_{a \in A_0} \{f(0) + m(a) + bv_{n+1}(1) + (1-b)v_{n+1}(0)\} && (4.30) \\ &= \min_{a \in A_0} \{m(a)\} + f(0) + bv_{n+1}(1) + (1-b)v_{n+1}(0) && (4.31) \end{aligned}$$

However, recall that since $m(a)$ is non-decreasing, a_1 will always be optimal at state $s = 0$. This is important since it allows our subsequent development to focus on the value function structure and optimal policy for states 1 and higher, avoiding inconvenient boundary conditions.

Next, we define convexity for functions on discrete sets, which is needed to prove optimality of a monotone policy. This definition is the discrete analogue of the condition that a differentiable function is convex if its first derivative is non-decreasing.

Definition 4.6. A function $g(x)$ defined on $\mathbb{Z}_+ = \{0, 1, \dots\}$ is *convex* if for all $x = 1, 2, \dots$

$$g(x+1) - g(x) \geq g(x) - g(x-1). \quad (4.32)$$

Alternatively, define $\Delta g(x) := g(x) - g(x-1)$. Then $g(x)$ is convex if

$$\Delta g(x+1) \geq \Delta g(x) \quad (4.33)$$

for all $x = 1, 2, \dots$.

Now, we proceed to showing optimality of a monotone policy. We do so under the assumption that the delay cost $f(s)$ and terminal reward $r_N(s)$ are convex functions of s. First, we show that under these assumptions, the value function $v_n(s)$ is convex in s for all $n = 1, \dots, N-1$ and $s = 1, 2, \dots$.

Lemma 4.2. Suppose $f(s)$ and $r_N(s)$ are convex functions of s. Then $v_n(s)$ is convex in s for $n = 1, \ldots, N$.

Proof. Let $s \in \{1, 2, \ldots\}$. We prove the result by induction on n. Clearly, the result is true at N since $v_N(s) = r_N(s)$ by definition. We will show that $\Delta v_n(s+1) \geq \Delta v_n(s)$ for all $n = 1, \ldots, N-1$. It suffices to show that $v_n(s)$ is convex assuming $v_{n+1}(s)$ is convex, due to induction and the assumption that $v_N(s)$ is convex.

Since $a^*_{n,s+1}$ need not attain the minimum at s

$$v_n(s) \leq f(s)+m(a^*_{n,s+1})+a^*_{n,s+1}v_{n+1}(s-1)+(1-a^*_{n,s+1}-b)v_{n+1}(s)+bv_{n+1}(s+1). \quad (4.34)$$

By definition of $a^*_{n,s+1}$, we have

$$v_n(s+1) = f(s+1)+m(a^*_{n,s+1})+a^*_{n,s+1}v_{n+1}(s)+(1-a^*_{n,s+1}-b)v_{n+1}(s+1)+bv_{n+1}(s+2)\}. \quad (4.35)$$

Subtracting (4.34) from (4.35) yields

$$\begin{aligned}\Delta v_n(s+1) &\geq \Delta f(s+1) + a^*_{n,s+1}\Delta v_{n+1}(s) + (1-a^*_{n,s+1}-b)\Delta v_{n+1}(s+1) + b\Delta v_{n+1}(s+2)\\ &\geq \Delta f(s+1) + a^*_{n,s+1}\Delta v_{n+1}(s) + (1-a^*_{n,s+1}-b)\Delta v_{n+1}(s) + b\Delta v_{n+1}(s+2)\\ &= \Delta f(s+1) + (1-b)\Delta v_{n+1}(s) + b\Delta v_{n+1}(s+2),\end{aligned}$$

where the second inequality follows from convexity of $v_{n+1}(s)$.

By a similar argument applied to states s and $s-1$,

$$\begin{aligned}\Delta v_n(s) &\leq \Delta f(s) + a^*_{n,s-1}\Delta v_{n+1}(s-1) + (1-a^*_{n,s-1}-b)\Delta v_{n+1}(s) + b\Delta v_{n+1}(s+1)\\ &\leq \Delta f(s) + (1-b)\Delta v_{n+1}(s) + b\Delta v_{n+1}(s+1).\end{aligned}$$

Combining the above inequalities, we have

$$\Delta v_n(s+1) - \Delta v_n(s) \geq \Delta f(s+1) - \Delta f(s) + b(\Delta v_{n+1}(s+2) - \Delta v_{n+1}(s+1)),$$

which is non-negative, due to convexity of $f(s)$ and $v_{n+1}(s)$.

□

Note that for a fixed action a, the value function is

$$v_n(s) = f(s) + m(a) + av_{n+1}(s-1) + (1-a-b)v_{n+1}(s) + bv_{n+1}(s+1).$$

Since the sum of convex functions is convex, it may be tempting to conclude that $v_n(s)$ is convex since the above equation holds for each $a \in A$, and so it must hold for an optimal a as well. However, the subtlety is that an optimal a depends on the state s, which is why our notation is of the form $a_{n,s}$. Thus, to follow this approach to prove convexity of $v_n(s)$ would require understanding how all the terms that involve $a_{n,s}$ vary as a function of s.

Theorem 4.3. Suppose $v_n(s)$ is a convex function of s for $n = 1, \dots, N-1$. Then $a^*_{n,s}$ is non-decreasing in s for $n = 1, \dots, N-1$.

Proof. Rewrite (4.28) as

$$v_n(s) = f(s) + b\Delta v_{n+1}(s+1) + v_{n+1}(s) + \min_{a \in A}\{m(a) - a\Delta v_{n+1}(s)\}. \tag{4.36}$$

Thus, it suffices to show that the smallest minimizer of $m(a) - a\Delta v_{n+1}(s)$ is non-decreasing in s.

Fix $s \in \{1, 2, \dots\}$. Let $a_{n,s}$ be the smallest minimizer of $m(a) - a\Delta v_{n+1}(s)$. Consider $s' > s$. We wish to show that $a_{n,s'} \geq a_{n,s}$. Assume to the contrary that $a_{n,s'} < a_{n,s}$. Then

$$\begin{aligned} m(a_{n,s'}) - a_{n,s'}\Delta v_{n+1}(s) &= m(a_{n,s'}) - a_{n,s'}\Delta v_{n+1}(s') + a_{n,s'}(\Delta v_{n+1}(s') - \Delta v_{n+1}(s)) \\ &\leq m(a_{n,s}) - a_{n,s}\Delta v_{n+1}(s') + a_{n,s'}(\Delta v_{n+1}(s') - \Delta v_{n+1}(s)) \\ &< m(a_{n,s}) - a_{n,s}\Delta v_{n+1}(s') + a_{n,s}(\Delta v_{n+1}(s') - \Delta v_{n+1}(s)) \\ &= m(a_{n,s}) - a_{n,s}\Delta v_{n+1}(s), \end{aligned}$$

where the first inequality is due to optimality of $a_{n,s'}$ and the second inequality is due to our assumption that $a'_{n,s} < a_{n,s}$ and convexity of $v_{n+1}(s)$. However, this contradicts the assumption that $a_{n,s}$ is the smallest minimizer of $m(a) - a\Delta v_{n+1}(s)$ so the result follows. □

Submodularity and Monotonicity

Note that we can arrive at this result a different way, using the concept of submodularity. We present this alternative approach as it is instructive and provides the reader with tools that can be used in other applications.

Definition 4.7. Let $h(x, y)$ be a real-valued function on $\mathbb{Z}^2_+$. Then $h(x, y)$ is *submodular* if

$$h(x^+, y^+) + h(x, y) \leq h(x^+, y) + h(x, y^+) \tag{4.37}$$

for $x, x^+, y, y^+ \in \mathbb{Z}_+, x^+ \geq x$ and $y^+ \geq y$.

Submodularity is useful in this case because of the following widely-used monotonicity result.

Proposition 4.2. Suppose $h(x, y)$ is submodular on $\mathbb{Z}^2_+$ and let

$$d(x) = \min\{y' \mid y' \in \arg\min_{y \in \mathbb{Z}_+} h(x, y)\}. \tag{4.38}$$

Then $d(x)$ is monotone non-decreasing on $\mathbb{Z}_+$.

Proof. Choose $x \in \mathbb{Z}_+$ and $x^+ \in \mathbb{Z}_+$ such that $x^+ \geq x$. We wish to show that $d(x^+) \geq d(x)$. For any $y \in \mathbb{Z}_+$ such that $y < d(x)$, the definition of $d(x)$ implies that

$$h(x, d(x)) - h(x, y) \leq 0. \tag{4.39}$$

Combining this inequality with (4.37), where we replace y^+ with $d(x)$, we have

$$\begin{aligned} h(x^+, d(x)) &\leq h(x^+, y) + h(x, d(x)) - h(x, y) \\ &\leq h(x^+, y). \end{aligned}$$

Since this inequality is true for all $y < d(x)$, any minimizer of $h(x^+, y)$ must be at least $d(x)$. So $d(x^+) \geq d(x)$. □

Two comments regarding this result follow:

1. To apply this result in a Markov decision process setting, x corresponds to a state and y corresponds to an action. Note that we have a finite action set, but the definition of submodularity above requires an infinite action set. To address this issue, we can simply define submodularity on a finite set, excluding the right boundary point.

2. Equation (4.38) accounts for multiple possible minimizers. If there is a unique minimizer, the expression simplifies to

$$d(x) = \operatorname*{arg\,min}_{y \in I_2} h(x, y).$$

The next result establishes submodularity of $m(a) + a\Delta v_{n+1}(s)$.

Lemma 4.3. Let $h(x, y) = m(y) - yg(x)$ where $m(y)$ is an arbitrary function of y and $g(x)$ is a non-decreasing in x. Then $h(x, y)$ is submodular.

Proof. Let $y^+ \geq y$. Since $g(x)$ is non-decreasing,

$$y(g(x^+) - g(x)) \leq y^+(g(x^+) - g(x)).$$

A little algebra shows that this inequality is equivalent to

$$m(y^+) - y^+g(x^+) + m(y) - yg(x) \leq m(y) - yg(x^+) + m(y^+) - y^+g(x),$$

which shows that $h(x, y)$ is submodular. □

With this additional machinery, we obtain the following concise proof of Theorem 4.3.

Alternative Proof of Theorem 4.3. By Lemma 4.3, $h(s,a) := m(a) + a\Delta v_{n+1}(s)$ is submodular. By Proposition 4.2, with $I_1 = S$ and $I_2 = A$, $a^*_{n,s}$ is non-decreasing in s. $\square$

We conclude this section with an example that illustrates the result. Suppose $A = \{a_0, a_1\}$, where $a_0 < a_1$. Then from the proof of Theorem 4.3, an optimal action at decision epoch n has the following simple form

$$a^*_s = \begin{cases} a_0 & \text{if } \Delta v_{n+1}(s) \le \bar{x} \\ a_1 & \text{if } \Delta v_{n+1}(s) > \bar{x}, \end{cases}$$

where $\bar{x} = (m(a_1) - m(a_0))/(a_1 - a_0)$.

4.4.3 An Optimal Policy for the Online Dating Problem

To investigate the structure of an optimal policy for the online dating problem from Section 3.8.2, we first write down the optimality equations and then solve a numerical example to obtain some insight.

Recall that the state space is $S = \{0, 1, \Delta\}$, where 1 indicates the current profile is the best encountered so far, 0 indicates that it is not, and Δ denotes the stopped state. Then $v_N(1) = 1$ and $v_N(0) = v_N(\Delta) = 0$.

For $n < N$,

$$v^*_n(0) = \max\left\{v^*_{n+1}(\Delta), \frac{1}{n+1}v^*_{n+1}(1) + \frac{n}{n+1}v^*_{n+1}(0)\right\}, \tag{4.40}$$

$$v^*_n(1) = \max\left\{\frac{n}{N} + v^*_{n+1}(\Delta), \frac{1}{n+1}v^*_{n+1}(1) + \frac{n}{n+1}v^*_{n+1}(0)\right\} \tag{4.41}$$

and $v^*_n(\Delta) = v^*_{n+1}(\Delta)$.

Since $v^*_N(\Delta) = 0$, $v^*_n(\Delta) = 0$ for all $n < N$. Noting that $v^*_n(s) \ge 0$ for $s = 0$ and 1, (4.40) and (4.41) reduce to

$$v^*_n(0) = \frac{1}{n+1}v^*_{n+1}(1) + \frac{n}{n+1}v^*_{n+1}(0) \tag{4.42}$$

and

$$v^*_n(1) = \max\left\{\frac{n}{N}, v^*_n(0)\right\} \tag{4.43}$$

In state 0, continuing is at least as good as stopping, so the optimal action is to continue. In state 1, an optimal action a^*_n is given by

$$a^*_n = \begin{cases} \text{stop} & \text{if } v^*_n(0) \le \frac{n}{N} \\ \text{continue} & \text{if } v^*_n(0) > \frac{n}{N}. \end{cases} \tag{4.44}$$

Note we adopt the convention that we stop when there is tie on the right hand side of (4.43). Equation (4.44) provides a specific structure for an optimal action as a function of the optimal value and the epoch. In the following example with four potential matches ($N = 4$), we examine how this structure translates into an optimal policy using equations (4.42) and (4.43).

Example 4.3. 1. Initialize calculations by setting $n = 4$ and $v_4^*(0) = 0$ and $v_4^*(1) = 1$.

2. Set $n = 3$ and

$$v_3^*(0) = \frac{1}{4}v_4^*(1) + \frac{3}{4}v_4^*(0) = \frac{1}{4} \tag{4.45}$$

and

$$v_3^*(1) = \max\left\{\frac{3}{4}, v_3^*(0)\right\} = \frac{3}{4}. \tag{4.46}$$

Hence $A_{3,1}^* = Q$ and it is optimal to stop at decision epoch 3 if the profile is the best so far.

3. Set $n = 2$ and

$$v_2^*(0) = \frac{1}{3}v_3^*(1) + \frac{2}{3}v_3^*(0) = \frac{1}{3}\cdot\frac{3}{4} + \frac{2}{3}\cdot\frac{1}{4} = \frac{5}{12} \tag{4.47}$$

and

$$v_2^*(1) = \max\left\{\frac{2}{4}, v_2^*(0)\right\} = \frac{2}{4} \tag{4.48}$$

Hence $A_{2,1}^* = Q$ and it is again optimal to stop at decision epoch 2 if the profile is the best so far.

4. Set $n = 1$ and

$$v_1^*(0) = \frac{1}{2}v_2^*(1) + \frac{1}{2}v_2^*(0) = \frac{1}{2}\cdot\frac{2}{4} + \frac{1}{2}\cdot\frac{5}{12} = \frac{11}{24} \tag{4.49}$$

and

$$v_2^*(1) = \max\left\{\frac{1}{4}, v_2^*(0)\right\} = \frac{11}{24} \tag{4.50}$$

Hence $A_{1,1}^* = C$ and it is optimal to continue at decision epoch 1.

This example shows that it is optimal to continue at the first decision epoch and then stop whenever a subsequent match is the best so far. The probability this strategy finds the best candidate is $\frac{11}{24} = 0.458$. If one repeats these calculations for larger values of N, similar optimal policies are observed where some number of initial matches are ignored and then the first profile that is the best observed so far is selected. Based on these observations, it is natural to hypothesize that the following policy is optimal for this problem: "Observe M profiles and then stop at the first subsequent profile is better than all that were previously viewed".

Theorem 4.4. Suppose $N > 2$. There exists a decision epoch $M \in \{1, \dots N-1\}$ for which an optimal policy is to choose action C (continue) for $n \leq M$ and then choose action Q (stop) at the first epoch $n > M$ for which $s = 1$. If $N > 2$, $1 \leq M < N$.

Proof. First, we show that if it is optimal to continue when $s = 1$ at decision epoch n' it is optimal to continue when $s = 1$ at decision epoch $n < n'$.

Since by hypothesis $v^*_{n'}(1) = v^*_{n'}(0)$,

$$v^*_{n'-1}(0) = \frac{1}{n'} v^*_{n'}(1) + \frac{n-1}{n'} v^*_{n'}(0) = v^*_{n'}(0) > \frac{n'}{N} > \frac{n'-1}{N}. \tag{4.51}$$

So by (4.44), it is optimal to continue when $s = 1$ at decision epoch $n' - 1$ and $v^*_{n'-1}(1) = v^*_{n'-1}(0)$. Repeating this argument shows that the result holds for all $n < n'$ and

$$v^*_1(1) = v^*_1(0) = v^*_2(1) = v^*_2(0) = \ldots = v^*_{n'}(1) = v^*_{n'}(0). \tag{4.52}$$

This establishes that the optimal policy cannot switch from continue to stop and then back to continue as a function of the epoch number.

To complete the proof, we must show that it is not optimal to always stop. Suppose $N > 2$. It follows easily from (4.41) that $M < N$. Now assume to the contrary that it is always optimal to stop. Then for all n, $v^*_n(1) = n/N$ and

$$v^*_n(0) = \frac{1}{n+1} \frac{n+1}{N} + \frac{n}{n+1} v^*_{n+1}(0) = \frac{1}{N} + \frac{n}{n+1} v^*_{n+1}(0). \tag{4.53}$$

Noting $v_N(0) = 0$ and applying (4.53) recursively we find that $v^*_{N-1}(0) = \frac{1}{N}$,

$$v^*_{N-2}(0) = \frac{N-2}{N} \left(\frac{1}{N-2} + \frac{1}{N-1} \right) \tag{4.54}$$

and in general

$$v^*_n(0) = \frac{n}{N} \left(\frac{1}{n} + \frac{1}{n+1} + \ldots + \frac{1}{N-1} \right). \tag{4.55}$$

Hence $v^*_1(0) > \frac{1}{N}$, which from (4.44) implies the optimal action at decision epoch 1 is to continue contradicting our assumption that it is always optimal to stop. Thus $M \geq 1$. □

The second part of the above proof suggests a convenient way to find an optimal policy. It is optimal to stop at epoch n if $v_n^*(0) \geq v_n^*(1)$, or equivalently if

$$\frac{1}{n} + \frac{1}{n+1} + \cdots + \frac{1}{N-1} \leq 1.$$

This yields the following important characterization of M:

$$M = \max\left\{ n \geq 1 \;\middle|\; \frac{1}{n} + \frac{1}{n+1} + \ldots + \frac{1}{N-1} > 1 \right\}. \tag{4.56}$$

Consistent with the numerical results in Example 4.3, equation (4.56) returns $M = 2$ when $N = 4$. If $N = 5$, then $M = 3$.

Equation (4.56) also leads to an elegant rule of thumb approximating an optimal policy, especially when N is large. Letting N become large and approximating the summation in (4.56) by an integral, we see that

$$\frac{1}{M} + \frac{1}{M+1} + \ldots + \frac{1}{N-1} \approx \int_M^N \frac{1}{x} dx = \ln \frac{N}{M}. \tag{4.57}$$

Thus, it is optimal to stop if

$$\ln \frac{N}{M} \approx 1 \text{ or } M \approx Ne^{-1} = 0.368N. \tag{4.58}$$

This result can be summarized succinctly as follows.

Observe 36.8% of the potential profiles and then choose the next one that is the best observed so far.

Combining (4.55) with (4.52) and (4.57) shows that

$$v_1^*(0) = v_1^*(1) = v_M^*(0) = \frac{Ne^{-1}}{N} = e^{-1} = 0.368.$$

This means that in the limit of large N, the probability the above policy finds the best match is 0.368.

Figure 4.5 plots M versus N. Notice that the relationship closely tracks the line $M = Ne^{-1}$, as suggested by the theory.

4.5 State-action value functions (q-functions)

The fundamental dynamic programming recursion

$$v_n^*(s) = \max_{a \in A_s} \left\{ \sum_{j \in S} p_n(j|s,a)(r_n(s,a,j) + v_{n+1}^*(j)) \right\} \tag{4.59}$$

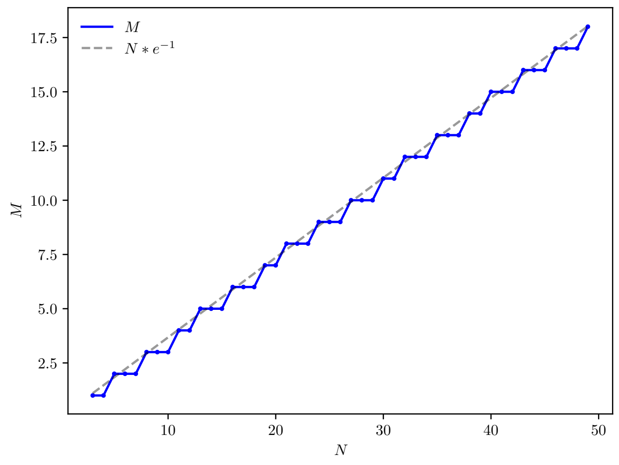

Figure 4.5: Graphical representation of the solution to the online dating problem.

may not be that useful when S is prohibitively large, because computing expectations for each action inside the maximum is time consuming. In addition, the transition probabilities and reward functions may not be known explicitly and may need to be determined by simulation or estimation from data. To address these challenges, we consider a slight modification of the decision-making perspective of our problem. Consider a period as running from immediately after selecting an action at one decision epoch until immediately after selecting an action at the next decision epoch. This is in contrast to the traditional viewpoint where the action is chosen at the start of a period, immediately after observing the state. See Figure 2.2 for a comparison of these two perspectives.

4.5.1 Recursions for q-functions

Define $q_n^\pi(s,a)$ to be the expected total reward from decision epoch n to the end of the planning horizon when action a is chosen in state s at decision epoch n, and the policy $\pi = (d_{n+1}, d_{n+2}, \dots, d_{N-1})$ is used subsequently. Then

$$q_{N-1}^\pi(s,a) = \sum_{j \in S} p_{N-1}(j|s,a)(r_{N-1}(s,a,j) + r_N(j)). \tag{4.60}$$

and for $n = 1, \dots, N-2$

$$q_n^\pi(s,a) = \sum_{j \in S} p_n(j|s,a)(r_n(s,a,j) + q_{n+1}^\pi(j, d_{n+1}(j))). \tag{4.61}$$

The advantage of representation (4.61) will become apparent when we optimize. The impact of the policy enters into (4.61) only through the appearance of d_{n+1} in the

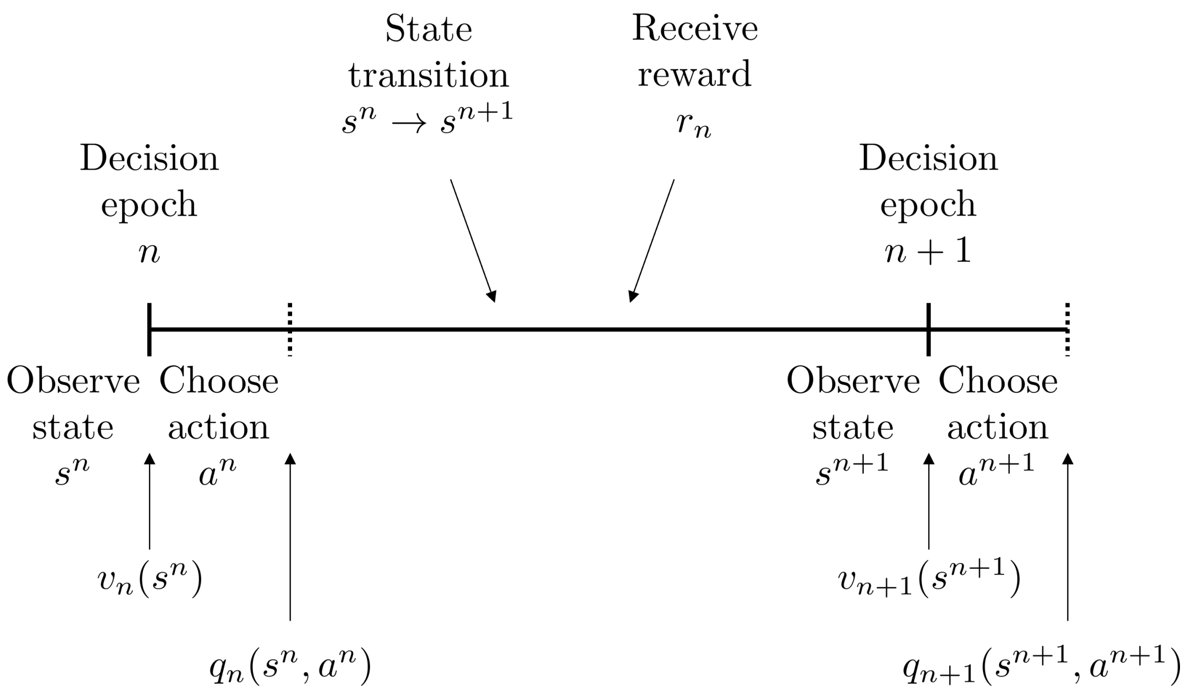

Figure 4.6: Timeline showing that $v_n(s^n)$ is computed after observing state s^n but before choosing action a^n, whereas $q_n(s^n, a^n)$ is computed after observing both s^n and a^n.

expression $q^{\pi}_{n+1}(j, d_{n+1}(j))$. Notice that (4.61) is similar to the expression for computing the value of a given policy $\pi = (d_1, d_2, \ldots, d_{N-1})$ from equation (4.6):

$$v^{\pi}_n(s) = \sum_{j \in S} p_n(j|s, d_n(s))(r_n(s, d_n(s), j) + v^{\pi}_{n+1}(j)) \tag{4.62}$$

The main difference is that there is only one action in (4.62), which is $d_n(s)$, whereas there are two actions present in (4.61), $d_n(s) = a$ and $d_{n+1}(j)$ where j is the next state after s. In other words computing the value function v requires two states and one action (state and action at epoch n and state at epoch $n + 1$), while q requires two states and two actions (states and actions at both epochs n and $n + 1$).

4.5.2 Revisiting Example 4.1 from the q-function perspective

Example 4.4. As in Example 4.1, let $N = 3$ and we evaluate the Markovian deterministic policy $\pi = (d_1, d_2)$, where

$$d_1(s_1) = a_{1,2}, \quad d_1(s_2) = a_{2,2}$$

and

$$d_2(s_1) = a_{1,1}, \quad d_2(s_2) = a_{2,1}.$$

When $n = 2$, we obtain:

$$\begin{aligned} q_2^\pi(s_1, a_{1,1}) &= p_2(s_1|s_1, a_{1,1})(r_2(s_1, a_{1,1}, s_1) + r_3(s_1)) + p_2(s_2|s_1, a_{1,1})(r_2(s_1, a_{1,1}, s_2) + r_3(s_2)) \\ &= 0.8(5 + 0) + 0.2(-5 + 0) \\ &= 3 \end{aligned}$$

and

$$\begin{aligned} q_2^\pi(s_2, a_{2,1}) &= p_2(s_1|s_2, a_{2,1})(r_2(s_2, a_{2,1}, s_1) + r_3(s_1)) + p_2(s_2|s_2, a_{2,1})(r_2(s_2, a_{2,1}, s_2) + r_3(s_2)) \\ &= 1(-5 + 0) \\ &= -5 \end{aligned}$$

When $n = 1$, we obtain:

$$\begin{aligned} q_1^\pi(s_1, a_{1,2}) = p_1(s_1|s_1, a_{1,2})(r_1(s_1, a_{1,2}, s_1) + q_2^\pi(s_1, d_2(s_1)))+& \\ p_1(s_2|s_1, a_{1,2})(r_1(s_1, a_{1,2}, s_2) + q_2^\pi(s_2, d_2(s_2)))& \quad (4.63) \end{aligned}$$

and

$$\begin{aligned} q_1^\pi(s_2, a_{2,2}) = p_1(s_1|s_2, a_{2,2})(r_1(s_2, a_{2,2}, s_1) + q_2^\pi(s_1, d_2(s_1)))+& \\ p_1(s_2|s_2, a_{2,2})(r_1(s_2, a_{2,2}, s_2) + q_2^\pi(s_2, d_2(s_2))).& \quad (4.64) \end{aligned}$$

Substituting values in (4.63) and (4.64) and noting that $d_2(s_1) = a_{1,1}$ and $d_2(s_2) = a_{2,1}$ yields

$$q_1^\pi(s_1, a_{1,2}) = 0(0 + 3) + 1(5 + (-5)) = 0$$

$$q_1^\pi(s_2, a_{2,2}) = 0.4(20 + 3) + 0.6(-10 + (-5)) = 0.2$$

Since $d_1(s_1) = a_{1,2}$ and $d_1(s_2) = a_{2,2}$ it follows that that $q_1^\pi(s_1, d_1(s_1)) = 0$ and $q_1^\pi(s_2, d_1(s_2)) = 0.2$.

Observe that:

1. These values agree with those in Section 4.1 and require the same computational effort. That is, $q_1^\pi(s_1, d_1(s_1)) = v_1^\pi(s_1) = 0$ and $q_1^\pi(s_2, d_1(s_2)) = v_1^\pi(s_2) = 0.2$.

2. There is no need to calculate q-values for actions that π does not use at decision epochs 1 and 2 because they do not enter into subsequent evaluations and are not required to determine $v_1^\pi(\cdot)$.

4.5.3 Computing optimal q-function Values

Simulation based methods in Chapters 11 and 12 focus on estimating "optimal" q-functions. Let

$$q_n^*(s,a) := \sup_{\pi \in \Pi^{\mathrm{HR}}} q_n^\pi(s,a) = \max_{\pi \in \Pi^{\mathrm{MD}}} q_n^\pi(s,a)$$

for $s \in S$ and $a \in A_s$, where the last equality can be proved following an analogous approach as the proof for Theorem 4.1.

Since there is no action to be taken at epoch N,

$$q_N^*(s,a) = r_N(s) \tag{4.65}$$

for all $s \in S$ and $a \in A_s$. In epoch $N-1$, we have

$$q_{N-1}^*(s,a) = \sum_{j \in S} p_{N-1}(j|s,a)(r_{N-1}(s,a,j) + r_N(j)). \tag{4.66}$$

In epoch $N-2$, the recursion needs to consider the decision to be made at the next epoch, $N-1$, which has to be made optimally

$$\begin{aligned} q_{N-2}^*(s,a) &= \max_{a' \in A_s} \left\{ \sum_{j \in S} p_{N-1}(j|s,a)(r_{N-1}(s,a,j) + q_{N-1}^*(j,a')) \right\} && (4.67) \\ &= \sum_{j \in S} p_{N-1}(j|s,a)(r_{N-1}(s,a,j) + \max_{a' \in A_s}\{q_{N-1}^*(j,a')\}). && (4.68) \end{aligned}$$

The second equality is due to the fact that a' only shows up in the q_{N-1}^* term. Continuing this development, and noting that $\max_{a \in A_s} q_N^*(s,a) = r_N(s)$, we can write the general recursion for $n = 1, \ldots, N-1$ as

$$q_n^*(s,a) = \sum_{j \in S} p_n(j|s,a)(r_n(s,a,j) + \max_{a' \in A_j}\{q_{n+1}^*(j,a')\}). \tag{4.69}$$

A few key observations about this development follow.

1. The recursion for $q_n^*(\cdot)$ differs from the recursion for $v_n^*(\cdot)$ in (4.22) in that the maximization is now inside the expectation. This change may not seem significant but it substantially reduces computation when $v_n^*(\cdot)$ is evaluated through simulation.

2. The expressions for q_{N-1}^π in (4.60) and q_{N-1}^* in (4.66) are identical because no decisions are made after action a is selected at decision epoch $N-1$.

3. There is a very close connection between v_n^* and q_n^*. In particular, for $n = 1, 2, ..., N$
$$v_n^*(s) = \max_{a \in A_s} q_n^*(s,a). \tag{4.70}$$

It then follows that

$$q_n^*(s,a) = \sum_{j \in S} p_n(j|s,a)(r_n(s,a,j) + v_{n+1}^*(j)). \tag{4.71}$$

Therefore $q_n^*(s,a)$ may be interpreted as *the expected total reward from decision epoch n onward when action a is chosen in state s and then the system evolves optimally.*

4. For $n = 1, 2, \ldots, N-1$ and $s \in S$, let $A_{n,s}^*$ denote the optimal actions in state s at decision epoch n. Previously, $A_{n,s}^*$ was defined in equation (4.21) using v_{n+1}^*:

$$A_{n,s}^* = \arg\max_{a \in A_s} \left\{ \sum_{j \in S} p_n(j|s,a)(r_n(s,a,j) + v_{n+1}^*(j)) \right\}.$$

Using q-functions, and noting equation (4.71), $A_{n,s}^*$ can be written much more succinctly as

$$A_{n,s}^* = \arg\max_{a \in A_s} q_n^*(s,a). \tag{4.72}$$

We found that when coding algorithms, q-functions provided a convenient intermediate step when evaluating determining optimal values and policies as shown in the following algorithm.

Algorithm 4.4: The Finite Horizon Policy Optimization Algorithm using q-functions

```
1 Set $n = N$, $q_N(s,a) = r_N(s)$ for all $s \in S$ and $a \in A_s$, and $d_N(s)$ to be an
    arbitrary action in $A_s$.
2 while $n > 1$ do
3     $n \leftarrow n - 1$
4     for $s \in S$ do
5         for $a \in A_s$ do
6             Evaluate $q_n(s,a)$ according to
                  $$q_n(s,a) = \sum_{j \in S} p_n(j|s,a)(r_n(s,a,j) + q_{n+1}(j, d_{n+1}(j))). \tag{4.73}$$
7         Set
                  $$A_{n,s}^* = \arg\max_{a \in A_s} q_n(s,a). \tag{4.74}$$
8         Select $d_n(s) \in A_{n,s}^*$.
9 return $q_1(s, d_1(s))$ for all $s \in S$ and $\pi = (d_1, \ldots, d_{N-1})$.
```

We leave it to the reader to establish that the computed q from Algorithm 4.4 is equal to q^*, in a similar way as that used to show that u from Algorithm 4.3 is equal to v^*,(Theorem 4.2).

4.5.4 Evaluating optimal q-functions and finding optimal policies

We now revisit Example 4.2 from the q-function perspective. To find q_1^* and q_2^*, we need to expand the previous calculations to account for all possible actions in each state at each decision epoch. We illustrate this calculation for $q_1^*(s_1, a_{1,1})$ and leave the remainder as an exercise.

$$\begin{aligned} q_1^*(s_1, a_{1,1}) &= p_1(s_1|s_1, a_{1,1})(r_1(s_1, a_{1,1}, s_1) + \max\{q_2^*(s_1, a_{1,1}), q_2^*(s_1, a_{1,2})\})+ \\ &\quad\ p_1(s_2|s_1, a_{1,1})(r_1(s_1, a_{1,1}, s_2) + \max\{q_2^*(s_2, a_{2,1}), q_2^*(s_2, a_{2,2})\}) \\ &= 0.8(5+5) + 0.2(-5+2) = 7.4 \end{aligned}$$

Table 4.2 gives all the possible values for $q_n^*(s, a)$.

State	Action	$q_2^*(s,a)$	$q_1^*(s,a)$
s_1	$a_{1,1}$	3	**7.4**
s_1	$a_{1,2}$	**5**	7
s_2	$a_{2,1}$	-5	-3
s_2	$a_{2,2}$	**2**	**5.2**

Table 4.2: Values of $q_n^*(s, a)$ for two-state model in Section 2.5. The maximum value in each state at each decision epoch is in **bold**.

It is easy to identify optimal policies and values from Table 4.2 by noting which actions attain the maximum at each decision epoch. These appear in Table 4.3.

Decision Epoch	State	Optimal Action	$v_n^*(s)$
1	s_1	$a_{1,1}$	7.4
1	s_2	$a_{2,2}$	5.2
2	s_1	$a_{2,2}$	5
2	s_2	$a_{2,2}$	2

Table 4.3: Optimal policies and values for two-state model in Section 2.5.

Inspecting Table 4.3 shows, for example, that

$$v_1^*(s_1) = \max\{q_1^*(s_1, a_{1,1}), q_1^*(s_1, a_{1,2})\} = 7.4.$$

and from (4.72) the optimal action in state s_1 at decision epoch 1 is given by

$$A_{1,s_1}^* = \arg\max\{q_1^*(s_1, a_{1,1}), q_1^*(s_1, a_{1,2})\} = a_{1,1}.$$

Observe that the optimal policies and values agree with those computed in Section 4.2.2 using corresponding algorithm for v^*.

4.5.5 Comparison of using q-functions and v-functions in the queuing admission control model

We consider a finite horizon version of the queuing admission control model of Section 3.3.2 and compare optimality equations expressed in terms of value functions $v(s)$ and state-action value functions $q(s,a)$. Recall that the state is represented by a vector (j,k) where the first component j denotes the number of jobs in the system and the second component k denotes whether there is a job waiting to be either admitted ($k=1$) or not ($k=0$).

For $j>0$, the optimality equations expressed in terms of value functions, $v(s)$, are

$$\begin{aligned} &v_n((j,1)) \\ &= \max\{R - h(j+1) + bv_{n+1}((j+1,1)) + (1-b-w)v_{n+1}((j+1,0)) + wv_{n+1}((j,0)), \\ &\qquad - h(j) + bv_{n+1}((j,1)) + (1-b-w)v_{n+1}((j,0)) + wv_{n+1}((j-1,0))\} \end{aligned} \tag{4.75}$$

and

$$v_n((j,0)) = -h(j) + bv_{n+1}((j,1)) + (1-b-w)v_{n+1}((j,0)) + wv_{n+1}((j-1,0)). \tag{4.76}$$

We leave it as an exercise to modify these equations for $j=0$. Observe that (4.76) does not contain a "max" since there is no decision when there is no one to admit.

We now consider the optimality equations based on the state-action value function, $q(s,a)$. Since we are basing the recursion on the *post-decision* state we do not need the second component of the state vector above. Recall that a_0 corresponds to "do not admit" and a_1 to "admit". Moreover the "do not admit" action encompasses two situations, when there is no arrival to admit and when there is an arrival to admit and the decision maker decides not to admit it. For $j>0$,

$$\begin{aligned} q_n(j,a_0) = -h(j) + (1-b-w)q_{n+1}(j,a_0) + wq_{n+1}(j-1,a_0) \\ + b\max\{q_{n+1}(j,a_0)), q_{n+1}(j,a_1)\} \end{aligned} \tag{4.77}$$

and

$$\begin{aligned} q_n(j,a_1) = R - h(j+1) + (1-b-w)q_{n+1}(j+1,a_0) + wq_{n+1}(j,a_0) \\ + b\max\{q_{n+1}(j+1,a_0)), q_{n+1}(j+1,a_1)\}. \end{aligned} \tag{4.78}$$

Again, we leave it as an exercise to modify equations (4.77) and (4.78) for $j=0$.

To understand the optimality equations for the state-action value function, notice that the terms inside the maximization have been considerably simplified. Since the state-action value function is based on the post-decision state, if there is no new arrival (the terms multiplied by $1-b-w$ and w), then there is no decision to be made (the action is a_0). Accordingly, the terms in the expected value calculation that involve transitions to a state where there is a single possible decision can be pulled outside of the max. Only if there is an arrival, which occurs with probability b, does the decision maker need to choose between a_0 and a_1 in the next epoch. The two terms inside the max reflect these two actions, whose values are entirely summarized by $q_{n+1}(s,a)$. Although this difference between q and v may seem minor, it has the effect of significantly simplifying computation in certain applications. See Exercise 22 at the end of this chapter.

Although q-functions simplify induction recursions here, the main advantage comes when analyzing the model with simulation. In that case (see Chapter 11) the simulated data would correspond to whether or not there was an arrival or service completion before the next decision epoch.

4.6 Technical Appendix

In this section, we show that the Finite Horizon Policy Optimization Algorithm finds optimal values and optimal policies within the class of Markovian deterministic policies. We then extend this result to show that these policies are optimal in the class of history-dependent policies.

Finding optimal policies in Π^{MD}

Proposition 4.3. Suppose that the maximum is attained in (4.22) for all $s \in S$, that $u_n(s)$ and $A^*_{n,s}$ for $n = 1, 2, \ldots, N-1$, $s \in S$ are computed by Algorithm 4.3 and $v^\pi_n(s)$ is as defined in equation (4.2). Then

1. For any $\pi \in \Pi^{\mathrm{MD}}$, $u_n(s) \geq v^\pi_n(s)$ for $n = 1, 2, \ldots, N$ and $s \in S$.
2. Suppose $\pi^* = (d^*_1, d^*_2, \ldots, d^*_{N-1})$ where $d^*_n(s) \in A^*_{n,s}$ for $n = 1, 2, \ldots, N-1$ and $s \in S$. Then $v^{\pi^*}_n(s) = u_n(s)$ for all $s \in S$ and $n = 1, 2, \ldots, N$.

Proof. We prove the first result by induction. Let $\pi = (d_1, d_2, \ldots, d_{N-1}) \in \Pi^{\mathrm{MD}}$. The result holds trivially for $n = N$ since $u_N(s) = r_N(s) = v^\pi_N(s)$ for all $s \in S$.

Assume now that $u_t(s) \geq v^\pi_t(s)$ for $t = n+1, \ldots, N$ and $s \in S$. From (4.22)

$$\begin{aligned} u_n(s) &= \max_{a \in A_s} \left\{ \sum_{j \in S} p_n(j|s,a)(r_n(s,a,j) + u_{n+1}(j)) \right\} \\ &\geq \sum_{j \in S} p_n(j|s,d_n(s))(r_n(s,d_n(s),j) + v^\pi_{n+1}(j)) \\ &= v^\pi_n(s), \end{aligned}$$

where the last equality holds from Proposition 4.1. Hence the result holds for $n = 1, \ldots, N$.

We prove the second result by induction as well. By the same argument as above, $u_N(s) = r_N(s) = v^{\pi^*}(s)$ for all $s \in S$. Now assume $v_t^{\pi^*}(s) = u_t(s)$ for all $s \in S$ and $t = n+1, \ldots, N$. By the definition of d_n^* together with (4.22) and (4.23) it follows for all $s \in S$ that

$$\begin{aligned} u_n(s) &= \max_{a \in A_s} \left\{ \sum_{j \in S} p_n(j|s,a)(r_n(s,a,j) + u_{n+1}(j)) \right\} && (4.79) \\ &= \sum_{j \in S} p_n(j|s, d_n^*(s))(r_n(s,a,d_n^*(s)) + u_{n+1}(j)) && (4.80) \\ &= \sum_{j \in S} p_n(j|s, d_n^*(s))(r_n(s,a,d_n^*(s)) + v_{n+1}^{\pi^*}(j)) && (4.81) \\ &= v_n^{\pi^*}(s), && (4.82) \end{aligned}$$

where the third equality follows from the induction hypothesis and the last equality follows from Proposition 4.1. So the result follows.

□

Corollary 4.2. Let π^* be as defined in Proposition 4.3. Then π^* is optimal in the class of Markov deterministic policies.

Finding optimal values in $\Pi^{\mathbf{HR}}$

Before proving that that there exist Markovian deterministic policies that are optimal in the class of all history-dependent randomized policies, we provide a conceptual algorithm for finding optimal values only.

Algorithm 4.5: The Finite Horizon Policy Optimization Algorithm in Π^{HR}

```
1 Set $n = N$ and $u'_N(h^N) = r_N(s^N)$ for all $h^N = (h^{N-1}, a^{N-1}, s^N) \in H_N$.
2 while $n > 1$ do
3     $n \leftarrow n - 1$
4     for $h^n = (h^{n-1}, a^{n-1}, s^n) \in H_n$ do
5         Evaluate $u'_n(h^n)$ according to
              $$u'_n(h^n) = \sup_{w \in \mathscr{P}(A_{s^n})} \left\{ \sum_{a \in A_{s^n}} w(a) \sum_{j \in S} p_n(j|s^n,a)(r_n(s^n,a,j) + u'_{n+1}((h^n,a,j)) \right\} \quad (4.83)$$
6 return $u'_1(s)$ for all $s \in S$ and $\pi = (d_1, \ldots, d_{N-1})$.
```

Note that (4.83) provides the most general form of the Bellman equations. A similar but slightly more tedious proof than that of Proposition 4.3 shows that this algorithm finds optimal values. We leave the proof as an exercise.

Proposition 4.4. Suppose for each $h^n \in H_n$, $n = 1, \dots, N$, $u'_n(h^n)$ is determined by Algorithm 4.5. Then $u'_n(h^n) = v^*_n(h^n)$ and $u'_1(s) = v^*(s)$ for all $s \in S$.

Optimality of Markov deterministic policies in $\Pi^{\mathbf{HR}}$

What remains to be shown is that in (4.83), a deterministic decision rule attains the "sup" over randomized decision rules and that u_n depends on $h^n = (h^{n-1}, a^n, s^n)$ only through s^n. We do this through two propositions. The first is an immediate consequence of the second part of Lemma 2.1 and its proof is omitted.

Proposition 4.5. Suppose for each $s \in S$, A_s is finite, and $u'_n(h^n), h^n \in H_N$ is computed by Algorithm 4.5. Then

$$\begin{aligned} u'_n(h^n) &= \sup_{w \in \mathscr{P}(A_{s^n})} \left\{ \sum_{a \in A_{s^n}} w(a) \sum_{j \in S} p_n(j|s^n, a)(r_n(s^n, a, j) + u'_{n+1}(h^{n+1})) \right\} \\ &= \max_{a \in A_{s^n}} \left\{ \sum_{j \in S} p_n(j|s^n, a)(r_n(s^n, a, j) + u'_{n+1}(h^{n+1})) \right\}. \end{aligned} \tag{4.84}$$

Proposition 4.6. Suppose for each $s \in S$, A_s is finite, $u_n(s)$ is computed by Algorithm 4.3, and $u'_n(h^n), h^n \in H_n$ is computed by Algorithm 4.5. Then for $n = 1, \dots, N-1$ and each $h^n = (h^{n-1}, a^n, s^n) \in H_n$, $u'_n(h^n)$ depends on h^n only through s^n. In other words, $u'_n(h^n) = u_n(s)$.

Proof. We prove this result by induction. By construction, $u'_N(h^N) = r_N(s^N) = u_N(s^N)$. Assume now that $u'_t(h^t) = u_t(s^t)$ for $t = n+1, \dots, N$. By the induction hypothesis and (4.84),

$$u'_n(h^n) = \max_{a \in A_{s^n}} \left\{ \sum_{j \in S} p_n(j|s^n, a)(r_n(s^n, a, j) + u_{n+1}(j)) \right\} \tag{4.85}$$

where $h^n = (h^{n-1}, a^n, s^n)$. Observe that all the quantities on the right hand side of (4.85) depend on h^n only through s^n. Hence, $u'_n(h^n) = u_n(s^n)$, the induction hypothesis holds and the result follows. □

The essence of Proposition 4.5 result is that randomization is not necessary when there are a finite number of actions. Proposition 4.6 states that only the current state matters if the transition probabilities, reward functions, and terminal reward do not depend on states prior to the current state. Together, these two results establish the following two equalities:

$$\sup_{\pi \in \Pi^{\mathrm{HR}}} v_n^\pi(h^n) = \max_{\pi \in \Pi^{\mathrm{HD}}} v_n^\pi(h^n) = \max_{\pi \in \Pi^{\mathrm{MD}}} v_n^\pi(s^n). \tag{4.86}$$

Thus, combining Propositions 4.4, 4.5 and 4.6 gives us Theorem 4.1. As a consequence of Theorem 4.1, a Markovian deterministic policy is optimal over the set of all history-dependent randomized policies. This justifies the restriction to Markovian deterministic policies at the beginning of this chapter. More formally

$$v^*(s) = \max_{\pi \in \Pi^{\mathrm{MD}}} v^\pi(s)$$

for all $s \in S$.

4.7 Bibliographic Remarks

The use of the backward induction to find optimal policies in a general Markov decision process model originates with Bellman [1957], where on page 87 he also states the *Principle of Optimality.* Precursors in specific application areas include Massé's work on reservoir management (Gessford and Karlin [1958]) and Wald's research on sequential analysis [Wald, 1947]. See Chapter 4 in Puterman [1994] for more on this background.

Derman [1970] provides the first rigorous treatment in book form of the existence of optimal Markovian deterministic policies in the class of history-dependent policies. His approach differs considerably from that in this chapter. His book is well worth reading for those interested in a concise exposition of Markov decision process theory.

For additional reading on relevant fundamentals in probability (e.g., nested conditional expectations) and queuing (e.g., steady state equations of the M/M/1 queue), the reader is referred to Feller [1982] and Kleinrock [1975].

The identification of structured optimal policies dates back at least to Scarf [1960], who proves the optimality of (s, S) policies in a periodic review inventory model with fixed ordering costs. Other examples include monotone optimal policies in queuing models, control limit policies in replacement problems and structured policies in clinical applications. Interpretability has become increasingly important in machine learning, see, for example, Rudin [2019].

Our proof of optimality of monotone policies in queuing service rate control follows Lippman [1975]. The use of submodularity in Markov decision processes originates with the elegant paper of Serfozo [1976]. For more on this concept, see Section 4.4.2 of Puterman [1994]. As noted in Chapter 3, the online dating problem originates with Cayley [1875]. Our analysis follows Bather [1980].

The concept of q-functions originates in the reinforcement learning literature, especially in the seminal paper by Watkins and Dayan [1992]. Our development follows Bertseksas [2012].

4.8 Exercises

1. Find an optimal policy for the model in Example 2.1 for $N = 2$ by enumerating and evaluating all Markovian deterministic policies using Algorithm 4.1.

2. Find an optimal policy for the model in Example 2.1 for $N = 4$ and $N = 5$ using Algorithm 4.3. For the same values of N, determine the smallest positive value of $r_N(s)$ for each s that would result in a different optimal policy.

3. Show that the values of $u_n(h^n)$ computed using Algorithm 4.2 are equal to $v_n^\pi(h^n)$ for all $h^n \in H_n$ and $n = 1, \ldots, N$.

4. Consider a model in which $S = \{s_0, s_1\}$, $A_{s_0} = \{a = 1, 2, \ldots\}$, $A_{s_1} = \{b\}$, $r(s_0, a) = 1 - \frac{1}{a}$, $r(s_1, b) = 0$, $p(s_1|s_0, a) = 1$ and $p(s_1|s_1, b) = 1$.

   (a) Represent the model graphically as in Figure 2.6.

   (b) Show that an optimal policy does not exist.

   (c) Show that for any $\varepsilon > 0$, there is a policy π^ε for which

   $$v^{\pi^\varepsilon}(s_0) \geq v^*(s_0) - \varepsilon.$$

5. Prove in general that for any $\varepsilon > 0$ there always exists an ε-optimal policy.

6. Consider the model proposed in Exercise 1 in Chapter 2.

   (a) Evaluate the expected total reward of the policy that uses action $a_{i,1}$ in each state in a 5-period model.

   (b) Find an optimal policy and its value in a 5-period version of this model.

   (c) Express the optimal equations in terms of q-functions.

7. For Example 2.1 with $N = 2$, use Algorithm 4.2 to evaluate the history-dependent randomized policy in Section 2.5.2 with $r_2(s_1) = r_2(s_2) = 0$ and parameter values $q_{1,1} = 0.8, q_{1,2} = 0.4$, $q_{2,1} = 0.1, q_{2,5} = 0.5, q_{2,2} = 0.2, q_{2,3} = 0.3, q_{2,4} = 0.4$ and $q_{2,6} = 0.6$

8. Solve the restaurant revenue management problem (Exercise 12 from Chapter 3) over a two-hour planning horizon. Clearly state and graphically represent the optimal policy.

9. Consider a 5-period version of the periodic inventory review problem of Section 3.2 with a finite warehouse of capacity of 10 units, no backlogging, a fixed ordering cost of 12, a per unit ordering cost of 2, revenue of 5 per item sold, and a holding cost of 1 per unit per period. Assume a scrap value of 1 per item. If demand cannot be fulfilled by stock on hand after receipt of an order, it is lost. Assume demand is geometrically distributed with $p = 0.3$.

(a) Solve the problem assuming there is no delay between when the order is placed and it arrives. Does the optimal policy have any obvious structure?

(b) Solve the problem assuming than when an order is placed and it arrives at the end of the period after demand is met from stock on hand.

10. Consider the lion hunting problem from Section 3.4 with a horizon of $N = 30$. Assume the lion only hunts gazelles. On a day it hunts, the lion catches a young gazelle with probability 0.2 and edible biomass of 8kg, an adult gazelle with probability 0.1 and edible biomass of 15kg, or nothing with probability 0.7. Assume the following additional parameter values: $C = 30, d = 6, h = 7, c_0 = 6$. Formulate and solve this problem.

11. Consider the network shown in Figure 4.7.

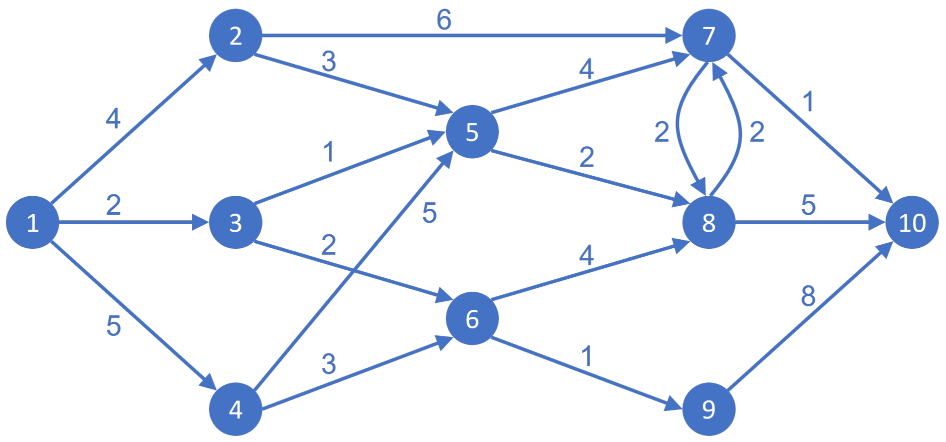

Figure 4.7: Network for Problem 11.

(a) Formulate the shortest path problem on this network as a finite-horizon Markov decision process where the states are the nodes and the actions are what arc to follow in each node. Identify all shortest paths from node 1 to node 10 using backward induction. Note that the use of periods in this example is artificial.

(b) This example has multiple shortest routes. Which would you prefer and why?

(c) Find the longest path from node 1 to node 10 using backward induction. In what application area might this be a useful problem to solve?

(d) Consider a modified network in which the arc from node 8 to node 7 has been deleted. Solve a stochastic version of this problem in which there is uncertainty about arc choice: if a node has two outgoing arcs, the chosen arc is followed with probability 0.8; if a node has three outgoing arcs, the chosen arc is followed with probability 0.6 and each of the remaining arcs is chosen with probability 0.2.

(e) Repeat the previous question but with the arc from node 8 to node 7 included.

12. Consider the "Pulling the goalie" model in Section 3.9.1.

    (a) Using the data at the end of that section, determine the optimal policy for when Team A should pull its goalie. Assume decisions are made every 20 seconds and that Team A first considers this decision when there are 5 minutes remaining.

    (b) Repeat this analysis for the situation when Team A's opponent, Team B, has a player in the penalty box.

13. Solve the single machine maintenance problem (Exercise 9 from Chapter 3), assuming it is a finite horizon problem with $N = 5$, using the following data: $r = 100$, $p(i) = (1 - e^{-i})/(1 + e^{-i})$, $c_A = 60$, $c_B = 20$.

14. Solve the house selling problem from Section 3.8.2 with $N = 7$, $f_n(s) = 0.001s$ for all n and $L_n(s) = 0.05s$ for all n. Assume the initial list price of the house is \$1M and that the best offer on the first day follows a discrete uniform distribution centered on the list price with a range of \$100K in \$10K increments. After the first day, we assume the best offer follows a similar discrete uniform distribution, but centered at the best offer from the previous day.

15. Consider the house selling problem in a hot housing market. The problem parameters are the same as in Exercise 14, except that the discrete uniform distribution of the best offer on day $n > 1$ is centered at the best offer seen so far since the start of the horizon.

16. Consider the house selling problem in an inflationary market. The central bank is considering a one-time interest rate increase, which will depress the offer values for the house by 5% compared to no rate increase. On each day, there is a probability 0.1 that interest rates will increase, as long as there was no previous increase. Modify the formulation accordingly and solve it using the parameters from Exercise 14.

17. Solve the queuing service rate control problem from Section 4.3.1 with the following changes to the parameters: a horizon length of $N = 6$, a delay cost of $f(s) = \sqrt{s}$ and a serving cost of $m(a) = 5a^3$. Comment on the monotonicity of the optimal policy compared to the policies depicted in Figure 4.3.

18. Prove that (4.56) holds by induction.

19. Prove that the sum of submodular functions is submodular.

20. Prove that if $g(x, y)$ does not depend on x, that is $g(x, y) = h(y)$ for some function of y, that $g(x, y)$ is submodular.

21. Consider the queuing admission control problem from Section 4.5.5. Derive the optimality equations for the v and q value functions for the case of $j = 0$.

22. Consider a finite horizon version of the queuing admission control problem from Section 4.5.5 with $N = 10$. Solve this problem using both the v and q value functions. Assume $b = 0.2$, $w = 0.2$, $h(j) = 5j$, and $R = 30$. At the final epoch, assume any remaining jobs in the system are penalized at a cost of 10 per job. Discuss the computational effort required for the two approaches.