Chapter 4

Finite Horizon Models

"Begin at the beginning," the King said gravely, "and go on till you come to the end: then stop."[1]

From *Alice's Adventures in Wonderland*, by Lewis Carroll, 1832-1898.

This chapter focuses on models in which the planning horizon, N, is finite and fixed, and decision makers evaluate and compare policies on the basis of their expected total reward. Such models will be referred to as *finite horizon models*.

The assumption of a *fixed* horizon length means that the decision maker knows the number of decision epochs in the planning horizon in advance. For example, in an airline reservation model, the date of the departure of a specific flight is fixed. This contrasts to situations in which the planning horizon is finite in expected value but its actual length may vary from realization to realization. Examples include clinical decision making problems (Section 3.6), which terminate at disease onset or death, or Gridworld models (Section 3.2) in which the robot's actions determine the time to complete a task. Models with variable horizon lengths are best analyzed with methods from Chapters 5 or 6 by converting a variable horizon length model into an infinite horizon model though the introduction of a zero-reward absorbing state.

Recall that a policy π is a sequence of decision rules $(d_1, d_2, \ldots, d_{N-1})$ that determines action choice at each decision epoch n, $n = 1, \ldots, N-1$, either deterministically or randomly. 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

[1] Carroll [1865].

holds in finite horizon models under mild assumptions, such as S finite and A_s finite for all $s \in S$. Thus, in this chapter, much of the exposition will focus on Markovian deterministic policies.

The results in this chapter, while important on their own, also provide a framework for analyzing and solving infinite horizon models in later chapters. In particular, this chapter introduces the following fundamental concepts:

1. Value of a policy,
2. Optimality equations,
3. Backwards induction (dynamic programming),
4. Existence of optimal policies,
5. Optimality of structured policies, and
6. Algorithms based on state-action value functions

The most significant results are that the existence of optimal policies within the class of Markovian deterministic policies and the development of a recursive algorithm for finding them.

4.1 The value of a policy in a finite horizon model

Definition 4.1. For each $s \in S$, the *value* of policy $\pi = (d_1, d_2, \ldots, d_{N-1}) \in \Pi^{\mathrm{HR}}$ is defined by

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

In (4.1) X_n denotes the system state and Y_n the action chosen (either deterministically or randomly) by π at decision epoch n. The quantity $v^\pi(s)$ is frequently referred to as a *policy value function.*

A fundamental insight in Markov decision process theory and computation is that evaluation of $v^\pi(s)$ can be accomplished by decomposing it into a series of one-period problems that can be evaluated recursively. Although (4.1) is valid for any type of policy, the equations for computing the value differ between policy classes. Section 4.1.2 discusses this recursion for a Markovian deterministic policy and Section 4.1.3 focuses on history-dependent randomized policies.

4.1.1 Backwards induction

This section describes a fundamental recursion underlying computation and analysis in a finite horizon model.

Definition 4.2. For each $s \in S$, define the *future expected reward from stage n* as

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

Observe that:

1. This quantity represents the expected total reward from decision epoch n to the end of the planning horizon using policy π, starting in state $s \in S$ at decision epoch n.
2. Alternatively, this quantity can be interpreted as the expected total reward in an $N-n$ period problem that starts at state $s \in S$.
3. This expression is similar to (4.1) except that the summation runs from decision epoch n through decision epoch $N-1$, and the subscript on the reward and decision rules agree with the decision epoch number.
4. When rewards correspond to costs, $v_n^\pi(s)$ is often referred to as the *cost-to-go*.
5. The expressions $v^\pi(s)$ and $v_1^\pi(s)$ represent the same quantity.

Recursions

This section motivates the basic recursion in finite horizon models by showing how $v_1^\pi(s)$ and $v_2^\pi(s)$ are related. It is easy to see that

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

The expression above shows that the value of policy $\pi = (d_1, \dots, d_{N-1})$ is the expected total reward obtained by using decision rule d_1 at decision epoch 1 in state s to generate action Y_1, plus the expected total reward from decision epoch 2 onward (to the end of the planning horizon) starting in state X_2. and using decision rules $d_2, \dots, d_{N-1}$. State X_2 is random because it is determined by transition probabilities and potentially random action selection that result from applying d_1 at decision epoch 1.

Since $v_2(\cdot)$ is unknown, forward recursions of this type cannot serve as the basis for computation. Rather, by starting at the end of the planning horizon and applying similar recursions backwards one obtains the following process that can be used for computation.

Begin by settting $v_N^\pi(s) = r_N(s)$ for all $s \in S$. Then set

$$\begin{aligned} v_{N-1}^\pi(s) &= E^\pi\left[r_{N-1}(X_{N-1}, Y_{N-1}, X_N) + r_N(X_N) \,\middle|\, X_{N-1} = s \right] \\ &= E^\pi\left[r_{N-1}(X_{N-1}, Y_{N-1}, X_N) + v_N^\pi(X_N) \,\middle|\, X_{N-1} = s \right]. \end{aligned} \tag{4.4}$$

Since $v_N^\pi(s)$ is know, the expectation in (4.4) can be evaluated. Continuing in this way, write $v_n^\pi(s)$ as

$$\begin{aligned} v_n^\pi(s) &= E^\pi\left[r_n(X_n, Y_n, X_{n+1}) + \sum_{i=n+1}^{N-1} r_i(X_i, Y_i, X_{i+1}) + r_N(X_N) \,\middle|\, X_n = s \right] \\ &= E^\pi\left[r_n(X_n, Y_n, X_{n+1}) + v_{n+1}^\pi(X_{n+1}) \,\middle|\, X_n = s \right]. \end{aligned} \tag{4.5}$$

for $n = 1, \dots, N-1$.

Once $v_{N-1}^\pi(s)$ for all $s \in S$ has been computed, use (4.5) to successively evaluate $v_{N-2}^\pi(s)$, $v_{N-3}^\pi(s)$ continuing all the way back to $v_1^\pi(s)$. Since $v_1^\pi(s) = v^\pi(s)$, this provides a way to find the value of policy π in every state. Such a computational approach is often referred to as *backwards induction* or sometimes as *dynamic programming*.

What backwards induction does is decompose the calculation of $v^\pi(s)$ into a series of calculations in one-period models like those described in Section 2.4. However the one-period models are linked through (4.5), where the quantity $v_{n+1}(s)$ is the terminal reward.

4.1.2 Evaluating a Markovian deterministic policy

This section provides an algorithm to compute the value of a Markovian deterministic policy and illustrates computations with an example. Since a Markovian deterministic policy generates a Markov reward process[2], the calculations also apply to any Markov reward process. As noted above, the fundamental idea is to decompose a $(N-1)$-period problem into a sequence of $N-1$ one-period problems.

Algorithm 4.1. Finite horizon policy evaluation for a Markovian deterministic policy

1. **Initialize:** Specify $\pi = (d_1, \dots, d_{N-1}) \in \Pi^{\text{MD}}$. Set $n \leftarrow N-1$ and $u_N(s) \leftarrow r_N(s)$ for all $s \in S$.

[2]Recall that a finite state Markov reward process is a Markov chain together with a reward function defined on the states of the chain.

```
2. **Iterate:** While $n \geq 1$:
    (a) For all $s \in S$, evaluate $u_n(s)$ according to
        $u_n(s) \leftarrow \sum_{j \in S} p_n(j|s, d_n(s))(r_n(s, d_n(s), j) + u_{n+1}(j)).$    (4.6)
    (b) $n \leftarrow n - 1$.
3. **Terminate:** Return $u_1(s)$ for all $s \in S$.
```

Some comments about this algorithm follow:

1. Theorem 4.1 below shows that this algorithm calculates the expected total reward of policy π for the entire planning horizon and also from each decision epoch onwards. Specifically, it shows that $u_n(s) = v_n^\pi(s)$ for $n = 1, \ldots, N-1$ and $s \in S$. Most importantly, $u_1(s) = v_1^\pi(s) = v^\pi(s)$.

2. This algorithm employs the dummy variables $u_n(s)$ to distinguish calculations in the algorithm from the definition of the value function, facilitating a formalization of their connection below.

3. When $|S| = M$, at each iteration of the algorithm, step 2(a) 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 runs for $N-1$ iterations, the total number of operations required is $O(NM^2)$. 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 only the values from the most recent iteration, then storage requirements reduce to $O(M)$. While this book will not focus extensively much on algorithmic efficiency, it is important to recognize that efficient implementation can significantly impact on computation.

4. If $r_n(s, a, j)$ does not depend on the subsequent state, equation (4.6) simplifies to
$$u_n(s) = r_n(s, d_n(s)) + \sum_{j \in S} p_n(j|s, d_n(s)) u_{n+1}(j). \tag{4.7}$$

5. When π is a Markovian randomized policy, one needs to add an additional expectation to take account of the randomness in action choice under d_n (see Section 4.1.3).

A numerical example

The following calculations compare two approaches for evaluating a Markovian deterministic policy in a finite horizon model:

1. Backwards induction
2. Sample path enumeration.

It does so in the context of the following instance of the two-state model from Section 2.5. Let $N = 3$, set $r_3(s_1) = r_3(s_2) = 0$ 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

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

Backwards induction: Explicit calculations using backwards induction as described in Algorithm 4.1 proceed as follows.

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

2. Set $n = 2$ and

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

and

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

3. Set $n = 1$ and

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

and

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

4. Since $n = 1$, stop.

Since $v^\pi(s) = v_1^\pi(s)$, the calculations above show that $v^\pi(s_1) = 0$ and $v^\pi(s_2) = 0.2$.

Sample path enumeration As described in Section 2.2.4, this requires determining the probability and value of each sample path. Suppose the process starts in state s_1. At the first decision epoch, d_1 chooses action $a_{1,2}$, which results in a deterministic transition to state s_2. At the second decision 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 the backwards induction calculations.

If the process starts in s_2, then several realizations are possible. They are summarized in Table 4.1.

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

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 $0.6(-15) + 0.32(25) + 0.08(15) = 0.2$, in agreement with the previous calculations.

For large N, the set of possible realizations becomes enormous, making sample path enumeration impractical for exact policy value function calculation. However, it forms the basis for the simulation methods in Chapter 10 so it is crucial to understand this concept.

Confirming that Algorithm 4.1 computes the value of a policy

This section proves that Algorithm 4.1 produces the value of a policy defined by equation (4.1). The proof uses the following basic result, which is sometimes referred to as the *Law of Iterated Expectations*[3].

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

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

[3]For a concrete example of this lemma, let X and Y be random variables that respectively denote the lifetime earnings and years of post-secondary education of a randomly chosen individual. Computing the average earnings of all individuals $E[X]$ 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 $E[X] = E[E[X|Y]] = \sum_{y=0}^{10} E[X|Y = y]P[Y = y]$.

Theorem 4.1. Let $\pi = (d_1, \ldots, d_{n-1}) \in \Pi^{\text{MD}}$. 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 as defined in equation (4.2). Then

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

Proof. The first result is proved by (backwards) 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 $t = n+1, \ldots, N$ 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,

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

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

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

This result holds as well for Markovian randomized policies. The proof requires only a few modifications and is left as an exercise.

4.1.3 Evaluating a history-dependent randomized policy*

You probably would not 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 within the class of history-dependent randomized policies. Exercise 8 asks you to use it to evaluate a history-dependent randomized policy in a simple example.

Let $\pi = (d_1, d_2, \ldots, d_{N-1}) \in \Pi^{\text{HR}}$ be a history-dependent randomized policy. This means that at decision epoch n, d_n chooses action a according to some probability distribution $w^n_{d_n}(a^n|h^n)$. 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 $v^\pi_n(h^n)$ 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. Finite horizon policy evaluation for a history-dependent randomized policy

1. **Initialize:** Specify $\pi = (d_1, \ldots, d_{N-1}) \in \Pi^{\text{HR}}$. Set $n \leftarrow N-1$ and $u_N(h^N) \leftarrow r_N(s^N)$ for all $h^N = (h^{N-1}, a^{N-1}, s^N) \in H_N$.
2. **Iterate:** While $n \geq 1$:
   (a) For all $h^n = (h^{n-1}, a^{n-1}, s^n) \in H_n$, evaluate $u_n(h^n)$ according to
   $$u_n(h^n) \leftarrow \sum_{j \in S} \sum_{a \in A_{s^n}} w^n_{d_n}(a|h^n) p_n(j|s^n, a)\big(r_n(s^n, a, j) + u_{n+1}(h^{n+1})\big), \tag{4.9}$$
   where $h^{n+1} = (h^n, a, j)$.
   (b) $n \leftarrow n - 1$.
3. **Terminate:** Return $u_1(h^1)$ for all $h^1 \in H_1$.

Some comments about Algorithm 4.2 follow:

1. It is left as an exercise to show that $u_n(h^n) = v^\pi_n(h^n)$ for all $h^n \in H_n$ and $n = 1, \ldots, N$.
2. In the first iteration (when $n = N$), the algorithm assigns the same value to all histories
   $$h^N = (s^1, a^1, s^2, \ldots, s^{N-1}, a^{N-1}, s^N)$$
   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 $w^n_{d_n}(a|h^n)$ 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 $w^n_{d_n}(a|h^n)$ 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

The goal in a finite horizon model is to find an optimal policy, that is, one that maximizes the expected total reward over all possible policies. This section, defines optimal policies and provides a recursion to compute them.

4.2.1 Definitions

Some key definitions follow.

Definition 4.3. A policy $\pi^* \in \Pi^{\mathrm{HR}}$ is an *optimal policy* if

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

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

Definition 4.4. 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, the optimal value from decision epoch n to the end of the planning horizon is given by

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

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 5 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 4.5. Suppose $\epsilon > 0$. Then a policy $\pi^\epsilon \in \Pi^{\text{HR}}$ is an *ϵ-optimal policy* if

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

for all $s \in S$.

Of course $v^*(s) \geq v^{\pi^\epsilon}(s)$ for all $s \in S$. Exercise 6 asks you to establish that for any Markov decision process, there always exists an ϵ-optimal policy.

The fundamental result

The following theorem states that under mild assumptions, Markovian deterministic policies are optimal within the broader class of history-dependent randomized policies. This means that the search for an optimal policy can be limited to the class of Markovian deterministic policies.

Theorem 4.2. 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,

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

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

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

This is a fundamental result within Markov decision processes theory that is often overlooked. A constructive proof is given below. Specifically, an algorithm is presented that generates value functions, which are subsequently proven to be optimal within the class of Markovian deterministic policies. Following this, it is shown that Markovian deterministic policies are optimal within the class of history-dependent randomized policies. Both of these proofs are provided in the chapter appendix.

Note that not only does Theorem 4.2 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*[4], to describe this property of optimal policies. He wrote [Bellman, 1957]:

[4]This result is true when maximizing the expected total reward but may not hold under other optimality criteria. See pp. 97-98 in Puterman [1994].

"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."

To understand why this must be true, consider driving from location A to Z along the shortest route, which involves visiting locations B, C, D, ..., Y on the way. Then it must be the case that the shortest route from B to Z visits C, D, ..., Y as well in the same order. The reason is that if there is a different route from B to Z that is shorter, then it could be used when driving from A to Z, contradicting the fact that the initial route from A to Z was shortest.

4.2.2 Computing optimal value functions and finding optimal policies

This section shows how to use backwards induction to compute $v^*(s)$ and use it to find an optimal policy, π^* when one exists. As a consequence of Theorem 4.2, 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

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

The quantity $v^*_{N-1}(s)$ represents the maximum expected total reward 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):

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

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 $s \in S$. Since 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 the above one-period problem.

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

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

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 $v^*_{N-1}(\cdot)$. Considering the interpretation of $v^*_{N-1}(s)$, $v^*_{N-2}(s)$ represents 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 $r_N(\cdot)$[5]. Repeating this argument leads to a recursive algorithm for computing $v^*(s)$ for all $\in S$.

Continuing in this way, define

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

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

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

Letting $d^*_n(s)$ be some $a^*_n \in A^*_{n,s}$ for each 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, $v^{\pi^*}(s) = v^*(s)$.

Backwards induction and the Bellman equation

The following algorithm computes an optimal value function and finds an optimal policy. This algorithm is often referred to as *Backwards Induction* or *Dynamic Programming*. Expression (4.22) is the fundamental relationship in finite horizon models. 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*[6].

Algorithm 4.3. Finite horizon policy optimization

1. **Initialize:** Set $n \leftarrow N-1$ and $u_N(s) \leftarrow r_N(s)$ for all $s \in S$.

2. **Iterate:** While $n \geq 1$:

   (a) For all $s \in S$, evaluate $u_n(s)$ according to

$$u_n(s) \leftarrow \max_{a\in A_s}\left\{\sum_{j\in S} p_n(j|s,a)\big(r_n(s,a,j)+u_{n+1}(j)\big)\right\}, \tag{4.22}$$

[5]This is the Principle of Optimality at work.

[6]Note that in a finite horizon model, the singular expression Bellman equation corresponds to an expression which holds for each n and all $s \in S$.

```
        set
```

$$A^*_{n,s} \leftarrow \arg\max_{a \in A_s} \left\{ \sum_{j \in S} p_n(j|s,a)\big(r_n(s,a,j) + u_{n+1}(j)\big) \right\} \tag{4.23}$$

```
        and select $d_n(s) \in A^*_{n,s}$.
    (b) $n \leftarrow n - 1$.
3. **Terminate:** Return $u_1(s)$ for all $s \in S$ and $\pi = (d_1, \ldots, d_{N-1})$.
```

Some comments about Algorithm 4.3 follow:

1. Theorem 4.3 below establishes that this algorithm determines that for all n, $v^*_n(s)$, 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 $s \in S$.

2. The sets $A^*_{n,s}$ may contain one or more elements. Finding this requires no additional computation. It is obtained when evaluating the maximum in (4.22) and storing those actions that achieve the maximum.

3. Any policy that chooses an action in $A^*_{n,s}$ for each $s \in S$ and $n = 1, \ldots, N-1$ is optimal. If this set contain a single element for all s and n, there is a unique optimal policy. However, when there are multiple optimal policies, one might be preferred to another because it has a more interpretable structure[7] or has smaller variance[8].

4. 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 $v^*_n(s)$.

5. Theorem 4.2 is applicable for both finite and countable state spaces. Accordingly, Algorithm 4.3 is applicable to both cases. However, one usually cannot calculate $u_n(s)$ for every s when 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

[7]Section 4.4 provides a systematic approach to identifying structured optimal policies.

[8]Recall, that $v^*_n(s)$ is an expected value. You may think of its variance as the variance of this value over multiple simulations or realizations.

threshold. The queuing service rate control application illustrates both options in Sections 4.3.1 and 4.4.3. Another option is to approximate $v_n^*(s)$ by a parametric function of s as in Chapter 9 but this does not guarantee finding an optimal policy.

The following theorem states that Algorithm 4.3 is guaranteed to find an optimal policy that is Markovian and deterministic, as well as the corresponding optimal values. The lengthy but insightful development needed to formally prove this result appears in the Appendix to this chapter. Combining Theorem 4.3 with Theorem 4.2 leads to the conclusion that that Algorithm 4.3 finds an optimal policy within the class of history-dependent randomized policies.

Theorem 4.3. 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

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

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

Returning to the discussion surrounding the Principle of Optimality, given Theorem 4.2, 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 assumption of finite action sets ensures 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. More formally, 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 $s \in S$ and $d_n^*(s) \in A_{n,s}^*$ for $n = 1, 2, \ldots, N-1$. Then the policy $\pi^* = (d_1^*, d_2^*, \ldots, d_{N-1}^*) \in \Pi^{\mathrm{MD}}$ is optimal.

4.2.3 A numerical example

The calculations below use Algorithm 4.3 to find an optimal policy and the corresponding optimal value in the two-state model from Section 2.5.

Example 4.1.

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 sets contain a single element at each decision epoch and state.

4.3 Applications

This section finds optimal policies in three of the applications from Chapter 3. In the first two applications, Algorithm 4.3 is used to numerically find optimal policies. An optimal policy is derived analytically in the third application.

Note that in the first two applications optimal policies are computed over a range of parameter values. The reason for doing this is to examine how the structure of the optimal policy varies as the model parameters change. We recommend that you carry out such *sensitivity analyses* in all applications.

4.3.1 Queuing service rate control

Consider a finite horizon variant of the service rate control problem described in Section 3.4.1. The calculations below examine how the optimal policy changes under two cost regimes (linear delay cost $f(s) = s$ and linear service cost $m(a) = 5a$; linear delay cost $f(s) = s$ and cubic service cost $m(a) = 5a^3$) and two horizon lengths ($N = 10$ and $N = 50$). Assume a terminal cost $r_N(s) = 0$ for all s, arrival probability $b = 0.1$ and three service probabilities $a_1 = 0.2$, $a_2 = 0.4$ and $a_3 = 0.6$.

Truncation

Calculations below use truncation to provide numerically tractable instances. To determine an appropriate truncation level W, steady state results for an $M/M/1$ queuing system[9] are used as follows.

Setting[10] the service rate to a and arrival rate to b ($b < a$), the mean queue length $\mu = b/(a-b)$ and the variance in queue length $\sigma^2 = ba/(a-b)^2$. Thus setting the truncation level

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

ensures that W is reached with low probability even under the lowest service probability. With the above parameter values, $\mu = 1$ and $\sigma = 1.4$ so that equation (4.25) implies that $W = 6$. Note that if $b = 0.199$ instead, then $\mu = 199$ and $W = 555$.

The Bellman equation

Since $m(a) + f(s)$ are costs it is more convenient to regard this as a cost minimization problem. To do so replaces the "max" in the Bellman equation by a "min" as follows:

$$u_n(s) = \min_{k\in\{1,2,3\}} \left\{ m(a_k) + f(s) + \sum_{j\in S} p(j|s,a_k)u_{n+1}(j) \right\} \tag{4.26}$$

for $n = 1, \dots, N-1$. Writing this out explicitly for each state gives

$$u_n(0) = \min_{k\in\{1,2,3\}} \{m(a_k) + f(0) + (1-b)u_{n+1}(0) + bu_{n+1}(1)\}$$

[9]An $M/M/1$ queuing system is a continuous time model for a single server queuing system with exponential inter-arrival times and service times.

[10]Using a and b as service and arrival rates, respectively, instead of probabilities is a reasonable approximation to the corresponding continuous time parameters for the chosen parameter values.

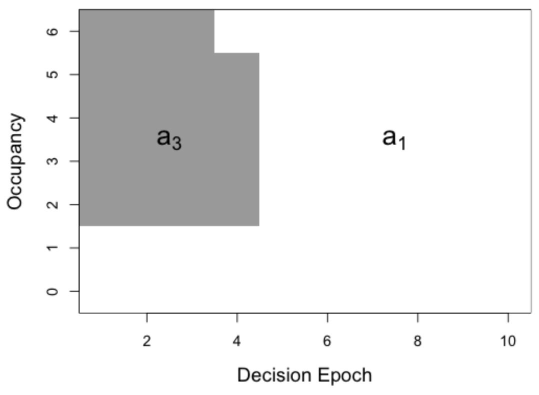
$$\begin{aligned} u_n(s) &= \min_{k\in\{1,2,3\}} \{m(a_k) + f(s) + a_k u_{n+1}(s-1) + (1-a_k-b)u_{n+1}(s) \\ &\qquad + b u_{n+1}(s+1)\}, \quad s = 1,\ldots,W-1 \\ u_n(W) &= \min_{k\in\{1,2,3\}} \{m(a_k) + f(W) + a_k u_{n+1}(W-1) + (1-a_k)u_{n+1}(W)\}. \end{aligned}$$

By assumption, $u_N(s) = r_N(s) = 0$ for all $s \in S$.

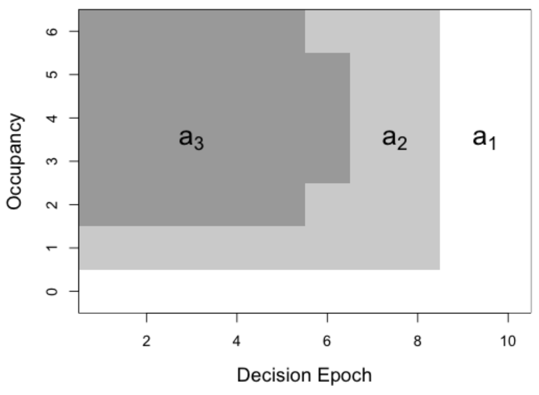
Note that it follows from the Bellman equation that in state 0 that the optimal service rate is always a_1 because $m(a)$ is increasing in a. This result is observed in all calculations.

Linear delay cost, linear service cost, $N = 10$

Figure 4.1a shows the optimal policy in each decision epoch for this instance.



(a) Linear delay cost and linear service cost



(b) Linear delay cost and cubic service cost

Figure 4.1: Optimal policy for queuing service rate control problem under two cost regimes and for $N = 10$. Occupancy refers to the state.

Observe the following structural properties of the optimal policy:

1. It is never optimal to use a_2.
2. The optimal policy at decision epochs 5 to 10 uses the least costly service rate a_1. This is because $r_{10}(s) = 0$, so there is no penalty for having the system occupied at termination.
3. In each state, the optimal service level decreases with respect to decision epoch. This again is a consequence of the end of planning horizon cost function.
4. The optimal action in state 0 is a_1 as noted above.

5. At all decision epochs except $n = 4$, the optimal policy is non-decreasing in the occupancy level. This reflects the reasonable assumption that the busier the system is, the faster the server should work. Theoretical justification for this observation is provided in Section 4.4.3 below.

6. The non-monotone decision rule at decision epoch 4 is an anomaly. It is a consequence of truncation of the state space at $s = 6$. The cost of using action a_1 (45.33) was close to that of using a_3 (45.35) so that if instead, the monotone policy was used at decision epoch 4, there would be a small increase in expected cost. More importantly when the truncation level $W > 6$, the optimal action in state 6 was a_3.

Linear delay cost, cubic service cost, $N = 10$

Figure 4.1b shows the optimal policy in each decision epoch for this instance. It shares the following properties with the linear service cost instance, namely:

1. It uses the least costly service rate in latter epochs.

2. It uses a_1 in state 0.

3. The service rate is non-increasing as a function of decision epoch.

4. The service rate is non-decreasing as a function of occupancy level (except at epoch 6).

5. The decision rule is a non-monotonic function of the occupancy at decision epoch 6.

Moreover the rationale for each are the same. However, the key difference with the optimal policy in the linear service cost case is that it uses a_2 in some states and decision epochs.

Linear delay cost, cubic service cost, $N = 50$

Figure 4.2 shows the optimal policy for this instance. The most important point to note is that the optimal policy remains the same up to decision epoch 46 at which point the terminal reward impacts actions choice, eventually using the slowest service rate.

Such behavior is described by what are know as *turnpike*[11] theorems.

[11]Turnpike theory is discussed in length in Section 6.8 in Puterman [1994]. The basic idea is starting from the end of the planning horizon, there exists a decision epoch after which the set of optimal policies remains the same.

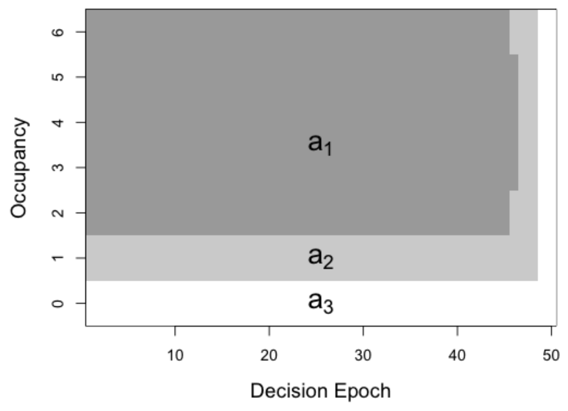


Figure 4.2: Optimal policy for queuing service rate control problem with linear delay cost, cubic service cost, and for $N = 50$. Occupancy refers to the state.

Conclusions

The figures above illustrate the optimal policies for each of three instances. The discussions that follow provide examples of the types of conclusions that can be drawn regarding the structure of an optimal policy. We encourage you try to draw similar conclusions when solving your own problems.

For these examples, truncation was applied to obtain numerical solutions. However this resulted in some unexpected "edge cases". To avoid such anomalies, when seeking solutions on a state space S, it is advisable to solve the problem on a larger state space $S' \supset S$ and then restrict the resulting policy to S.

It is also noteworthy that although there were three actions available, the optimal policy in the linear service cost case only used the two extreme actions a_1 and a_3. Such policies that abruptly switch between extreme actions are often referred to as *bang-bang*[12] *policies.* One can prove that when costs are linear only extreme actions are optimal.

4.3.2 Revenue management

This section computes optimal policies in an example that is naturally represented by a finite horizon model: the revenue management problem in Section 3.3.

Assume the selling season is $N = 6$ months and the retailer has $M = 15$ units of inventory to sell. The candidate prices are $A_s = \{20, 23, 25, 27, 30, 35\}$ for $s \in \{1, \dots, M\}$ and there are two possible scrap values, $H = 0$ or $H = 5$, allowing investigation of the

[12]This refers to a policy that switches from the "smallest" action to the "largest" or vice versa, assuming the actions are ordered. When there are only two actions, this is equivalent to a control-limit or threshold policy.

sensitivity of the optimal policy to the scrap value. The holding cost is calculated based on the end of month inventory item and is set to 2 per item so that $h(j) = 2j$. Monthly demand is Poisson with rate $\lambda_{n,a} = (1.1 - 0.1n)(9 - 0.25a)$, which is linearly decreasing in both a and n. As a consequence, $0.25 \leq \lambda_{n,a} \leq 4$ for $n = 1$ and $0.15 \leq \lambda_{n,a} \leq 2.4$ for $n = 5$.

Bellman equation

The Bellman equations for this model are $u_N(s) = Hs$ for $s = 0, \dots, M$,

$$u_n(s) = \max_{a \in A_s} \left\{ \sum_{j=0}^{s-1} \left(\frac{e^{-\lambda_{n,a}} \lambda_{n,a}^j}{j!} \big(aj - h(s-j) + u_{n+1}(s-j) \big) \right) + \left(\sum_{j=s}^{\infty} \frac{e^{-\lambda_{n,a}} \lambda_{n,a}^j}{j!} \right) \big(as + u_{n+1}(0) \big) \right\}$$

for $s = 1, \dots, M$ and $u_n(0) = 0$ for $n = 1, \dots, N-1$. Since $u_n(0) = 0$, the above expression simplifies to

$$u_n(s) = \max_{a \in A_s} \left\{ \sum_{j=0}^{s-1} \left(\frac{e^{-\lambda_{n,a}} \lambda_{n,a}^j}{j!} \big(aj - h(s-j) + u_{n+1}(s-j) \big) \right) + as \sum_{j=s}^{\infty} \frac{e^{-\lambda_{n,a}} \lambda_{n,a}^j}{j!} \right\}.$$

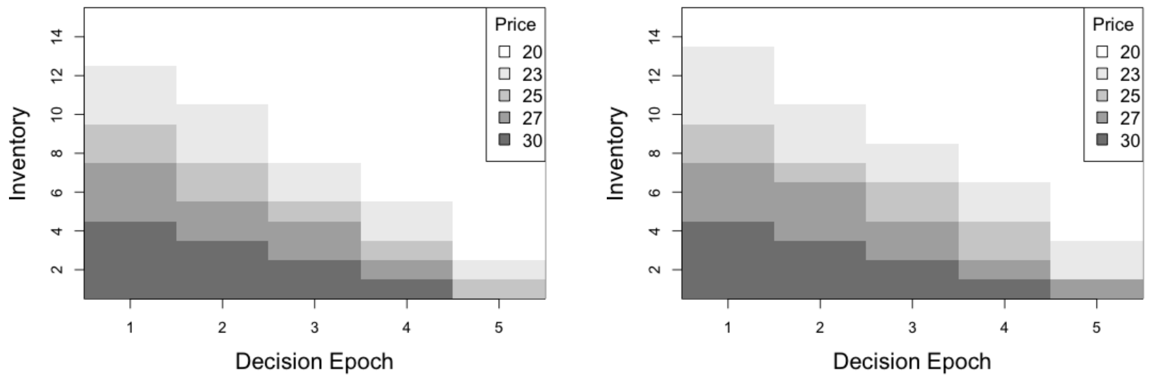
Note the first summation in the "max" corresponds to the possibility that demand j falls below the inventory level s, so the remaining inventory is $s - j$, while the second term corresponds to the possibility that demand equals or exceeds the inventory level. In the latter case, the number of items sold at price a is the number of items in inventory. Consequently the inventory at the start of the next decision epoch is zero.

Results

The following is the result of applying Algorithm 4.3 to this example. Figure 4.3 depicts optimal policies for the two scrap values. For example at decision epoch 2, with $H = 0$, the optimal prices are given by

$$d_2^*(s) = \begin{cases} 30 & s = 1, 2, 3 \\ 27 & s = 4, 5 \\ 25 & s = 6, 7 \\ 23 & s = 8, 9, 10 \\ 20 & s = 11, \dots, 15. \end{cases}$$

Note that for both scrap values, it is never optimal to set the price equal to 35. This is because of the low mean demand at this price which varies between 0.25 and 0.15.



(a) Scrap value $H = 0$

(b) Scrap value $H = 5$

Figure 4.3: Optimal policy for revenue management example. Grey tones represent the different optimal prices.

Observe for each scrap value, the optimal policies have the same general shape. At each decision epoch, optimal prices are non-increasing in the inventory level. This means that in each decision epoch, it is optimal for the retailer to retain the current price or set a lower price when the inventory level is higher. Conversely, as the inventory level decreases it is optimal for the price to increase or remain the same. For a fixed inventory level, the price is non-increasing over time. This means that as the end of the season nears it is optimal to markdown the price or keep it the same.

With a higher scrap value $H = 5$, the optimal decision rule will set prices to be the same or higher at each inventory level and decision epoch compared to when with $H = 0$. For example, with $H = 5$, $d_5^*(3) = 23$ and $d_5^*(1) = 27$ while when $H = 0$, $d_5^*(3) = 20$ and $d_5^*(1) = 25$. The reason for this is that when the scrap value is higher, the retailer can take on some of the risk of lower demand at higher prices.

The results of these calculations agree with what is observed in practice: prices for fashion goods decline over time, but do so more slowly when fewer items are available.

Fixed pricing

One might ask, "What is the benefit of using dynamic pricing?" This can be investigated by using the policy evaluation equations with the policy set equal to a fixed price. At price $\bar{a}$ they are given by $u_N(s) = Hs$,

$$u_n(s) = \sum_{j=0}^{s-1} \left(\frac{e^{-\lambda_{n,\bar{a}}} \lambda_{n,\bar{a}}^j}{j!} \big(\bar{a}j - h(s-j) + u_{n+1}(s-j) \big) \right) + \bar{a}s \sum_{j=s}^{\infty} \frac{e^{-\lambda_{n,\bar{a}}} \lambda_{n,\bar{a}}^j}{j!}$$

for $n = 1, \ldots, N-1$ and $s = 1, \ldots, M$ and $u_n(0) = 0$ for $n = 1, \ldots, N$. Note that this expression is equivalent to choosing $A_s = \{\bar{a}\}$ for $s = 1, \ldots, M$ so that the same code used to maximize the expected total reward can be used by simply modifying A_s accordingly.

Table 4.2 compares the expected total reward over 5 periods using a fixed price to that using the optimal policy. Observe that the optimal policy outperforms all specified fixed price policies for both salvage values. Moreover observe that the expected total reward is negative when the price is set to 35. This is a result of both the low demand and holding costs of 2 dollars per period.

Note the expected total revenue is slightly higher than that obtained using the fixed price of 20. This is especially consequential in retail inventory management where profit margins are low and a large number of products are sold.

Fixed price ($\bar{a}$)	$H = 0$	$H = 5$
18	215.58	218.35
20	225.16	230.77
23	216.05	229.43
25	190.52	211.60
30	68.97	113.98
35	−108.50	−38.50
Optimal policy	230.65	237.55

Table 4.2: Comparison of the expected total reward obtained using a range of fixed prices denoted by $\bar{a}$ with that of the optimal dynamic policy in the revenue management example with salvage values $H = 0$ and $H = 5$ and a starting inventory of 15 units.

4.3.3 Online dating: Finding the best match

This section uses the Bellman equations to derive the optimal policy *analytically*, instead of numerically, in the online dating[13] problem formulated in Section 3.8.2. As noted in the formulation, the objective in this model is to find a policy that maximizes the *probability* of obtaining the best match.

In this model, 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. When $s = 0$ or $s = 1$, $A_s = \{a_0, a_1\}$ where a_0 represents not choosing the current match and a_1 denotes choosing the current match. Also $A_\Delta = \{a_0\}$, meaning that once the stopped state is entered there are no further decisions to make[14].

The Bellman equations follow. One can interpret the quantity $u_n(s)$ as the probability of obtaining the best match given that the decision process is in state s at decision epoch n. Clearly $u_N(1) = 1$ and $u_N(0) = u_N(\Delta) = 0$. For $n < N$,

$$u_n(0) = \max\left\{u_{n+1}(\Delta),\, \frac{1}{n+1}u_{n+1}(1) + \frac{n}{n+1}u_{n+1}(0)\right\}, \tag{4.27}$$

[13] As previously noted, this example is typically referred to as the *secretary problem.*

[14] The reward is received only at epoch when the decision to stop is implemented.

$$u_n(1) = \max\left\{\frac{n}{N} + u_{n+1}(\Delta),\ \frac{1}{n+1}u_{n+1}(1) + \frac{n}{n+1}u_{n+1}(0)\right\} \tag{4.28}$$

and $u_n(\Delta) = u_{n+1}(\Delta)$.

Since $u_N(\Delta) = 0$ and Δ is an absorbing state[15], $u_n(\Delta) = 0$ for all $n < N$. Noting that $u_n(s) \geq 0$ for $s = 0$ and 1, or arguing that it is never optimal to stop if the current match is not the best so far, (4.27) and (4.28) reduce to:

$$u_n(0) = \frac{1}{n+1}u_{n+1}(1) + \frac{n}{n+1}u_{n+1}(0) \tag{4.29}$$

and

$$u_n(1) = \max\left\{\frac{n}{N}, u_n(0)\right\}. \tag{4.30}$$

Of course one could have derived these equations directly.

Thus in state 0, one should pass on the profile and in state 1, an optimal decision rule (or action a_n^*) is given by

$$d_n^*(1) = a_n^* = \begin{cases} \text{stop} & \text{if } u_n(0) \leq n/N \\ \text{continue} & \text{if } u_n(0) > n/N. \end{cases} \tag{4.31}$$

The above expression assumes the convention that it is optimal to accept the current profile when there is a tie on the right hand side of (4.30). Equation (4.31) provides a specific structure for an optimal action as a function of the optimal value and the epoch. However, to implement it requires knowing $u_n(0)$.

An example

The following example with four potential matches ($N = 4$), illustrates the computation of an optimal policy using equations (4.29) and (4.30).

1. Initialize calculations by setting $n = 4$ and $u_4(0) = 0$ and $u_4(1) = 1$. This makes sense since if the last profile is the best so far, it is the best overall and if not, one cannot obtain the best match.

2. Set $n = 3$,

$$u_3(0) = \frac{1}{4}u_4(1) + \frac{3}{4}u_4(0) = \frac{1}{4} \tag{4.32}$$

and

$$u_3(1) = \max\left\{\frac{3}{4}, u_3(0)\right\} = \frac{3}{4}. \tag{4.33}$$

[15] See Appendix B for the definition of this concept.

Hence the set of optimal actions, $A^*_{3,1} = a_1$, and it is optimal to stop at decision epoch 3 if the profile is the best so far. Moreover the probability of getting the best match if in state 0 is 1/4 (which occurs only if the last profile is the best) and 3/4 by stopping in state 1 since the likelihood the last profile is the best is $1 - 3/4$.

3. Set $n = 2$ and

$$u_2(0) = \frac{1}{3}u_3(1) + \frac{2}{3}u_3(0) = \frac{1}{3} \cdot \frac{3}{4} + \frac{2}{3} \cdot \frac{1}{4} = \frac{5}{12} \tag{4.34}$$

and

$$u_2(1) = \max\left\{\frac{2}{4}, u_2(0)\right\} = \frac{2}{4} \tag{4.35}$$

Hence $A^*_{2,1} = a_1$ and it is again optimal to stop at decision epoch 2 if the profile is the best so far. This makes sense since if the state is 1 and one continues, the probability of getting the best match is 5/12 while if one stops, it is 1/2.

4. Set $n = 1$ and

$$u_1(0) = \frac{1}{2}u_2(1) + \frac{1}{2}u_2(0) = \frac{1}{2} \cdot \frac{2}{4} + \frac{1}{2} \cdot \frac{5}{12} = \frac{11}{24} \tag{4.36}$$

and

$$u_2(1) = \max\left\{\frac{1}{4}, u_2(0)\right\} = \frac{11}{24} \tag{4.37}$$

Hence $A^*_{1,1} = a_0$ and it is optimal to continue at decision epoch 1. Again this makes sense since by default the first match is the best so far and the probability it is the best match overall is 1/4 and the probability of achieving the best match by continuing is 11/24.

Thus the optimal policy for this example is

$$d^*_n(0) = a_0 \text{ for } n = 1, 2, 3 \quad \text{and} \quad d^*_n(1) = \begin{cases} a_0 \text{ for } n = 1 \\ a_1 \text{ for } n = 2, 3. \end{cases}$$

It can be interpreted as continue at the first decision epoch and then stop whenever a subsequent match is the best observed so far. The probability that this strategy finds the best candidate is $11/24 = 0.458$.

The optimal policy

Repeating the above calculations for larger values of N reveals that the optimal policy rejects any profile for an initial period of length M after which it selects the first profile that is better than all of those observed so far. Of course, if the online dater does not choose a profile before the last one, then the last one must be chosen.

The following theorem demonstrates that, indeed, the optimal policy has this structure.

Theorem 4.4. Suppose $N > 2$. Then there exists a decision epoch $M \in \{1, \dots N-1\}$ for which the optimal policy $\pi^* = (d_1^*, \dots, d_{N-1}^*)$ satisfies:

$$d_n^*(0) = a_0 \text{ for } 1 \leq n \leq N-1 \quad \text{and} \quad d_n^*(1) = \begin{cases} a_0 \text{ for } 1 \leq n \leq M \\ a_1 \text{ for } M+1 \leq n \leq N-1. \end{cases} \tag{4.38}$$

Proof. First, the proof shows that if it is optimal to continue when $s = 1$ at decision epoch n' it is optimal to continue when $s = 1$ at all earlier decision epochs $n < n'$.

Assuming it is optimal to continue at n', it follows from (4.30) that $u_{n'}(1) = u_{n'}(0)$ so that

$$u_{n'-1}(0) = \frac{1}{n'} u_{n'}(1) + \frac{n-1}{n'} u_{n'}(0) = u_{n'}(0) > \frac{n'}{N} > \frac{n'-1}{N}. \tag{4.39}$$

So by (4.31), it is optimal to continue when $s = 1$ at decision epoch $n'-1$ and $u_{n'-1}(1) = u_{n'-1}(0)$. Repeating this argument shows that the result holds for all $n < n'$ and

$$u_1(1) = u_1(0) = u_2(1) = u_2(0) = \dots = u_{n'}(1) = u_{n'}(0). \tag{4.40}$$

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

The proof is completed by showing that it is not optimal to always stop, or equivalently for some n, $u_n(0) > n/N$. Suppose $N > 2$. Now assume to the contrary that it is always optimal to stop. Then for all n, $u_n(1) = n/N$ and

$$u_n(0) = \frac{1}{n+1} \frac{n+1}{N} + \frac{n}{n+1} u_{n+1}(0) = \frac{1}{N} + \frac{n}{n+1} u_{n+1}(0). \tag{4.41}$$

Noting $u_N(0) = 0$ and applying (4.41) recursively shows that $u_{N-1}(0) = 1/N$,

$$u_{N-2}(0) = \frac{N-2}{N} \left(\frac{1}{N-2} + \frac{1}{N-1} \right) \tag{4.42}$$

and in general under this assumption

$$u_n(0) = \frac{n}{N} \left(\frac{1}{n} + \frac{1}{n+1} + \dots + \frac{1}{N-1} \right). \tag{4.43}$$

Hence it is easy to see $u_1(0) > 1/N$, which from (4.31) implies that the optimal action at decision epoch 1 is to continue, contradicting the assumption that it is always optimal to stop. Thus $N > 1$. Since one must stop when $n = N$, $M < N$. □

Some comments regarding the conclusions and hypothesis of the theorem follow:

1. It states that the optimal policy is to choose action a_0 (continue) for $n \leq M$ and then choose action a_1 (stop) at the first epoch $n > M$ for which the current match is better than all previously observed matches.

2. Using this policy it is possible that one might not obtain the best match. This occurs if the best profile is among the first M.

3. The theorem excludes the case $N = 2$, since the probability of getting the best match is $1/2$ regardless of whether the online dater selects the first match or waits until the second since the two rankings $(1, 2)$ and $(2, 1)$ are equally likely.

An approximation to M

The second part of the proof above suggests a convenient way to find an optimal policy. It implies that it is optimal to stop at epoch n if $u_n(0) \geq u_n(1)$, 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:

$$M = \max\left\{ n \geq 1 \,\middle|\, \frac{1}{n} + \frac{1}{n+1} + \ldots + \frac{1}{N-1} > 1 \right\}. \tag{4.44}$$

Note that is consistent with the numerical results above since (4.44) returns $M = 1$ when $N = 4$. This means that it is optimal to pass over the first profile and then choose any profile that is better than all of those previously seen. Note that if $N = 5$, $M = 2$.

Equation (4.44) leads to the following elegant rule of thumb for choosing M when N is large. In this case the summation in (4.44) can be approximated by an integral as follows:

$$\frac{1}{M} + \frac{1}{M+1} + \ldots + \frac{1}{N-1} \approx \int_M^N \frac{1}{x} dx = \ln \frac{N}{M}. \tag{4.45}$$

Thus, it is optimal to stop if

$$\ln \frac{N}{M} \approx 1 \quad \text{or equivalently,} \qquad M \approx N e^{-1} = 0.368N. \tag{4.46}$$

This result can be summarized succinctly as follows.

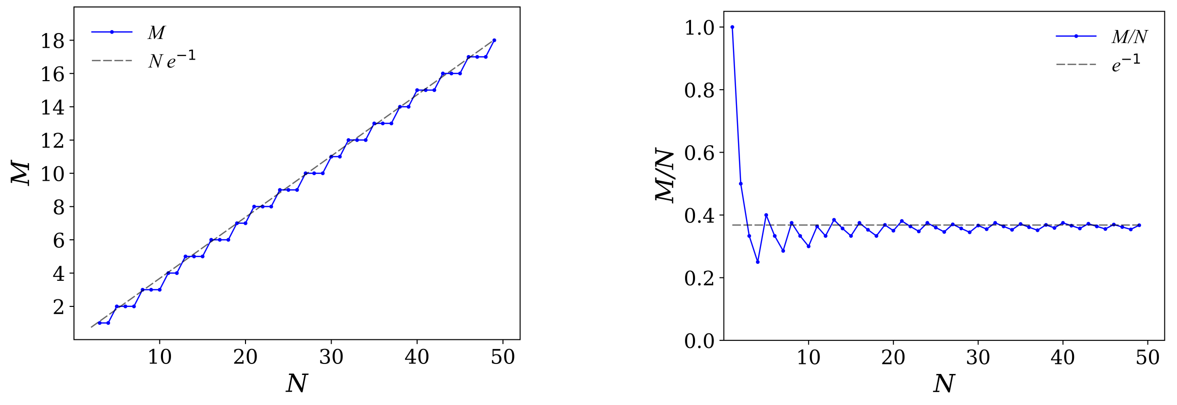
Observe 36.8% of the potential profiles and then choose the next one that is the best observed so far.

Combining (4.43) with (4.40) and (4.45) shows that

$$u_1(0) = u_1(1) = u_M(0) = \frac{Ne^{-1}}{N} = e^{-1} = 0.368.$$

This means that for large N, the optimal policy finds the best match with probability 0.368 or 36.8% of the time.

Figure 4.4a plots M versus N. Notice that the relationship closely tracks the line $M = Ne^{-1}$, as suggested by the theory. Figure 4.4b plots M/N versus N. This figure clearly shows how M/N converges to e^{-1}, the probability that the optimal policy finds the best match.



(a) Comparing M to Ne^{-1}. Since Ne^{-1} closely approximates M, stopping after viewing Ne^{-1} profiles is a good approximation to the optimal policy.

(b) The function M/N as a function of N. As N increases, M/N converges rapidly to e^{-1}, the probability that the optimal policy finds the best match.

Figure 4.4: Graphical representation of the optimal solution to the online dating problem and its approximation.

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 dating 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. Scarf [1960] showed that there exists an (s, S) policy (Figure 1.1) within the set of optimal policies. Such a policy is implemented as follows: if at any decision epoch the inventory level falls below s, an order is placed to raise the inventory level up to S. Such policies are also referred to as *min-max* policies and are implemented in many commercial inventory systems. Policies of this form are easy to understand and implement. The primary challenge is to determine optimal values for the quantities s and S.

In addition to providing a 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 policy 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.

The online dating problem above provides another instance of a model in which the optimal policy has an interpretable structure. The analysis in Section 4.3.3 shows that it is optimal for the online dater to observe the first M profiles and then choose the first subsequent profile better than all of those previously observed. By doing so, the online dater is able to establish a benchmark on which to base decisions following decision epoch M. Moreover, knowing that the optimal policy has this form, one can use an approximation to determine M.

4.4.1 A general approach

Assume for simplicity that S is ordered[16], for example $S = \{0, 1, \dots, M\}$. Let V^F denote the set[17] of real-valued functions of a specified form (such as convex) on S and let D^F be a subset of D^{MD} in which all decision rules have a specified structure. The sets D^F and V^F are *compatible* if for every $v(\cdot)$ in V^F and $s \in S$, there exists a

$$d(s) \in \operatorname*{arg\,max}_{a \in A_s} \left\{ \sum_{p \in S} p_n(j|s, a)\big(r_n(s, a, j) + v(j)\big) \right\} \tag{4.47}$$

in D^F for $n = 1, 2, \dots, N-1$.

Analyses below establish that:

1. In a machine maintenance model, non-decreasing value functions are compatible with control-limit policies, and

2. in a queuing service rate control model, convex non-decreasing value functions are compatible with monotone non-decreasing Markovian deterministic decision rules.

The following theorem provides a framework for establishing the optimality of a structured policy. It relies heavily on backwards induction.

[16]These analyses generalize easily to partially ordered S such as $\mathbb{R}^n$.

[17]Subsequent chapters will express the Bellman equation as an operator on a set of value functions.

Theorem 4.5. Suppose that $r_N(\cdot)$ is in V^F and for each $v(\cdot)$ in V^F, $v'(\cdot)$ defined by

$$v'(s) = \max_{a \in A_s} \left\{ \sum_{p \in S} p_n(j|s,a)\big(r_n(s,a,j) + v(j)\big) \right\} \tag{4.48}$$

for all $s \in S$ is in V^F for $n = 1, 2, \dots, N-1$.

Then

1. $v_n^*(s)$ is in V^F for $n = 1, 2, \dots, N-1$.
2. If D^F is compatible with V^F, there exists a Markovian deterministic optimal policy
$$\pi^* = \big(d_1^*(s), \dots, d_{N-1}^*(s)\big)$$
with $d_n^*(\cdot) \in D^F$ for $n = 1, \dots, N-1$.

Proof. The proof is by backwards induction. Since $v_N^*(s) = r_N(s)$ for all $s \in S$, $v_N^*(.)$ is in V^F. Next assume that for $t = N-1, N-2, \dots, n+1$, $v_t^*(\cdot)$ is in V^F. Then by (4.48) $v_n^*(\cdot)$ is in V^F. Hence $v_n^*(s)$ is in V^F for $n = 1, 2, \dots, N-1$. Thus, if D^F is compatible with V^F, $d_n^*(\cdot)$ is in D^F for $n = 1, 2, \dots, N-1$. □

By providing the necessary ingredients to establish the optimality of a structured policy, this result avoids the need to recreate the induction argument in every application. The challenge when applying it is to verify (4.48).

4.4.2 Example: Preventive maintenance

This section illustrates the use of Theorem 4.5 by applying it to determine the structure of an optimal policy in a simple dynamic machine maintenance model. In it, a machine deteriorates probabilistically over time and becomes more costly to operate as its state deteriorates. Alternatively, the machine can be restored to a pristine state at any time at a fixed cost.

Model formulation

Let $S = \{0, 1, \dots, M\}$ represent the state of a machine; a higher state value represents greater deterioration. Assume that the state of the machine is known at the beginning of period n, at which time the decision maker may either restore it (action a_1) to state 0 or operate it as is (action a_0). Hence $A_s = \{a_0, a_1\}$ for $s \in S$.

Restoring the machine in period n costs K_n and operating the machine in state s for one-period, incurs an operating cost $f_n(s)$ for $n = 1, 2, \dots, N-1$. Assume that $f_n(s)$ is a non-decreasing function of s for all $n = 1, \dots, N$. This means that operating the machine in a more deteriorated state is at least as costly as operating it in a less deteriorated state[18].

If the machine is not restored (action $a_0 0$) in period n and is in state $s < M$, it deteriorates to state $s+1$ with probability p_n and remains in state s with probability $1 - p_n$. Furthermore assume the machine remains in state M, unless it is restored. Restoring the machine (action a_1) requires one period resulting in a machine in state 0 at the start of the next period.

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

$$p_n(j|s,a) = \begin{cases} p_n & j = s+1, a = a_0, s \leq M-1 \\ 1 - p_n & j = s, a = a_0, s \leq M-1 \\ 1 & j = 0, a = a_1, s \leq M \quad \text{or} \quad j = M, a = 0, s = M \\ 0 & \text{otherwise.} \end{cases}$$

Since the decision maker seeks to minimize the expected cost of operating the machine over N periods, this problem is best expressed as a minimization problem. The Bellman equations for the model are

$$u_n(s) = f_N(s)$$

for $s \in S$ and

$$u_n(s) = \begin{cases} \min\left\{K_n + u_{n+1}(0), f_n(s) + p_n v_{n+1}(s+1) + (1-p_n)u_{n+1}(s)\right\} & 0 \leq s \leq M-1 \\ \min\{K_n + u_{n+1}(0), f_n(M) + u_{n+1}(M)\} & s = M \end{cases} \tag{4.49}$$

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

Control-limit policies are optimal

To apply Theorem 4.5 requires specifying V^F and showing that when $v(\cdot)$ is in V^F, so is $v'(\cdot)$ as defined by (4.48).

Let V^F denote the set of non-decreasing functions on S. By assumption $f_N(\cdot)$ is in V^F. Now suppose $v(s)$ is non-decreasing in s. Then

$$f_n(s+1) + p_n v(s+2) + (1-p_n)v(s+1) - \big(f_n(s) + p_n v(s+1) + (1-p_n)v(s)\big) \geq 0$$

for $s \leq M-2$ and

$$f_n(M) + v(M) - \big(f_n(s) + p_n v(M) + (1-p_n)v(M-1)\big) \geq 0,$$

so that the second expression in (4.49) is non-decreasing.

Since $K_n + v(0)$ is a constant, it is non-decreasing. Noting that the minimum of two non-decreasing functions is non-decreasing, establishes that $v'(s)$ is non-decreasing and

[18]This cost might be the result of producing lower quality output and reducing revenue.

in V^F. Hence the hypotheses of Theorem 4.5 are satisfied and $v_n^*(s)$ is non-decreasing in s

What remains is to determine the form of D^F. At decision epoch n, for $0 \leq s \leq M-1$,

$$d_n(s) = \begin{cases} 0 & \text{if } f_n(s) + p_n v(s+1) + (1-p_n)v(s) \leq K_n + v(0) \\ 1 & \text{if } f_n(s) + p_n v(s+1) + (1-p_n)v(s) \geq K_n + v(0) \end{cases} \tag{4.50}$$

and

$$d_n(M) = \begin{cases} 0 & \text{if } f_n(M) + v(M) \leq K_n + v(0) \\ 1 & \text{if } f_n(M) + v(M) \geq K_n + v(0). \end{cases} \tag{4.51}$$

For any non-decreasing $v(\cdot)$, $f_n(s) + p_n v(s+1) + (1-p_n)v(s)$ crosses $K_n + v(0)$ at most once so it follows from (4.50) and (4.51)[19] that the set of compatible decision rules has the form

$$d_n(s) = \begin{cases} 0 & \text{if } s \leq s_n \\ 1 & \text{if } s \geq s_n \end{cases} \tag{4.52}$$

for some constants s_n. Figure 4.5 shows how s_n is determined.

Such a policy is referred to as a *control-limit* or *threshold* policy and the quantity s_n denotes the control-limit or threshold at decision epoch n. Thus D^F represents the set of decision rules of form (4.52) establishing the following result.

Theorem 4.6. Suppose $f_n(s)$ is non-decreasing for $n = 1, 2, \ldots, n$. Then there exists an optimal control-limit policy in the machine maintenance model.

Figure 4.5 shows that it is possible that in (4.52)[20] that $s_n < 0$ meaning that it is optimal to restore the machine in every state in period n or $s_n > M$ meaning it is optimal to not restore the machine in period n. Moreover, if $s = s_n$, both actions are optimal.

It is left as an exercise to provide conditions on the model parameters that ensure $s_n \in \{0, 1, \ldots, M\}$ and also to determine an easy way to compute it. Furthermore one might be interested in determining how s_n behaves as a function of n.

4.4.3 Monotonicity of optimal policies in service rate control queuing systems*

Section 4.3.1 showed optimal policies exhibited the following interpretable structures:

[19] It is left to the reader to complete the argument for $s = M$.

[20] After extending the function beyond $\{0, \ldots, M\}$.

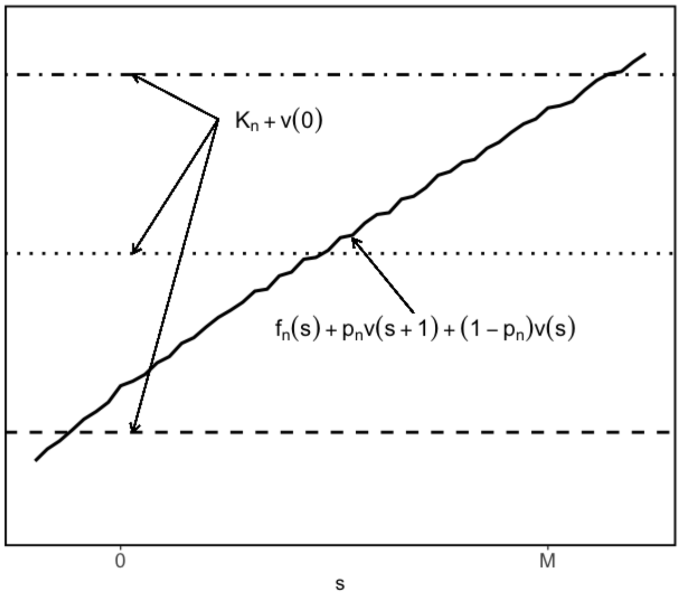


Figure 4.5: Graphical representation of calculation of s_n. It is the value at which the non-decreasing function (extended beyond $\{0, \dots, M\}$) crosses one of the horizontal lines corresponding to possible values for $K_n + v(0)$. There are three cases; $s_n > M$, ($K_n + v(0)$ denoted by dash-dot line), $0 < s_n < M$ (dotted line) and $0 > s_n$ (dashed line). Note the computations must be adjusted for the comparison of $f(M) + v(M)$ to $K_n + v(0)$.

1. When delay costs and holding costs were linear the optimal policy chose either the lowest service probability a_1 or the highest service probability a_3. Furthermore, except at one decision epoch when truncation resulted in non-monotone behavior at the boundary, the optimal policy exhibited a "bang-bang" structure. That is, there was a particular state s' such that it chose the slowest service rate a_1 when $0 \leq s \leq s'$ and the fastest service rate a_3 when $s' < s \leq W$.

2. In another parameter regime with linear delay costs and cubic serving costs, the optimal policy selected all three actions, with larger values of the action being optimal in larger state values. Again at one decision epoch, state-space truncation introduced non-monotone behavior.

3. In all cases and in all states, the optimal policy was monotone in the decision epoch.

This section formalizes the first two observations theoretically. It shows that under reasonable conditions, an optimal policy for the service rate control problem is a monotone policy. That is, assuming ordered states and actions, the optimal action a_s^* is non-decreasing in s. To avoid the non-monotonicity due to state space truncation, this section assumes a *countably infinite* state space $S = \{0, 1, \dots\}$.

Definition 4.6. A decision rule $d(s)$ is *monotone* if $d(s)$ is non-decreasing or non-increasing in s. A policy $\pi = (d_1, d_2, \ldots, d_N)$ is monotone if it consists of monotone decision rules $d_1, d_2, \ldots, d_N$.

The Bellman equation

For $n = 1, \ldots, N-1$ and $s = 1, 2, \ldots$ the Bellman equation for this model, written in terms of cost minimization, is given by

$$u_n(s) = \min_{a \in A_s}\{f(s) + m(a) + au_{n+1}(s-1) + (1-a-b)u_{n+1}(s) + bu_{n+1}(s+1)\} \quad (4.53)$$

and for $s = 0$ by

$$u_n(0) = \min_{a \in A_0}\{f(0) + m(a) + bu_{n+1}(1) + (1-b)u_{n+1}(0)\}. \quad (4.54)$$

Moreover $u_N(s) = r_N(s)$ for all $s \in S$.

Given a solution $v_n^*(s)$ to this system of equations subject to the boundary condition, define $A_{n,s}^*$ by

$$A_{n,s}^* = \arg\min_{a \in A}\{f(s) + m(a) + av_{n+1}^*(s-1) + (1-a-b)v_{n+1}^*(s) + bv_{n+1}^*(s+1)\}. \quad (4.55)$$

Since this set may have multiple elements, define $a_{n,s}^* = \min\{a \mid a \in A_{n,s}^*\}$ to be the smallest element in the set.

Note that when $s = 0$, (4.54) reduces to

$$u_n(0) = \min_{a \in A_0}\{m(a)\} + f(0) + bu_{n+1}(1) + (1-b)u_{n+1}(0).$$

Since $m(a)$ is non-decreasing, a_1 is the optimal action at state $s = 0$ for all n, or in other words, $A_{n,0}^* = \{a_1\}$ for $n = 1, \ldots, N-1$. This intuitive observation is important since it allows the subsequent development to focus on the value function structure and optimal policy for states 1 and higher, avoiding inconvenient boundary conditions at $s = 0$.

The main result

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

Definition 4.7. A function $g(x)$ defined on $\mathbb{Z}_+ = \{0, 1, \ldots\}$ is *convex* if for all $x = 1, 2, \ldots$

$$g(x+1) - g(x) \geq g(x) - g(x-1). \tag{4.56}$$

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

$$\Delta g(x+1) \geq \Delta g(x) \tag{4.57}$$

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

The following arguments establish optimality of a monotone policy by first establishing the convexity of solution of the Bellman equations under the assumption that the delay cost $f(s)$ and terminal reward $r_N(s)$ are convex functions of s.

Lemma 4.2. Suppose $f(s)$ and $r_N(s)$ are convex functions of s and let $v_n^*(s)$ be solutions of (4.53) and (4.54) subject to $u_n(s) = r_N(s)$ for all $s \in S$.
Then $v_n^*(s)$ is convex in s for $n = 1, \ldots, N$.

Proof. Let $s \in \{1, 2, \ldots\}$. The proof proceeds by (backwards) induction on n.

Clearly, the result holds for $n = N$ since $v_N^*(s) = r_N(s)$ by definition. Now assume that $v_{n+1}^*(s)$ is convex. Under this assumption, the following arguments establish that $\Delta v_n^*(s+1) \geq \Delta v_n^*(s)$ for $s = 1, 2, \ldots$.

Since $a_{n,s+1}^*$ need not attain the minimum at s, it follows from (4.53) that

$$v_n^*(s) \leq f(s) + m(a_{n,s+1}^*) + a_{n,s+1}^* v_{n+1}^*(s-1) + (1 - a_{n,s+1}^* - b) v_{n+1}^*(s) + b v_{n+1}^*(s+1). \tag{4.58}$$

By the definition of $a_{n,s+1}^*$,

$$v_n^*(s+1) = f(s+1) + m(a_{n,s+1}^*) + a_{n,s+1}^* v_{n+1}^*(s) + (1 - a_{n,s+1}^* - b) v_{n+1}^*(s+1) + b v_{n+1}^*(s+2)\}. \tag{4.59}$$

Subtracting (4.58) from (4.59) 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 at s and $s-1$,

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

where the convexity of $v^*_{n+1}(s)$ implies inequality $\Delta v^*_{n+1}(s) \geq \Delta v^*_{n+1}(s-1)$ resulting in the inequality above.

Combining the above inequalities, gives

$$\Delta v^*_n(s+1) - \Delta v^*_n(s) \geq \Delta f(s+1) - \Delta f(s) + b(\Delta v^*_{n+1}(s+2) - \Delta v^*_{n+1}(s+1)),$$

which is non-negative, due to convexity of $f(s)$ and $v^*_{n+1}(s)$. Hence $v^*_n(s)$ is convex. Thus the induction hypothesis is satisfied from which the convexity of $v^*_n(s)$ for $n = 1, 2, \ldots, N$ follows.

□

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

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

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 $a \in A_s$ and consequently would hold for an optimal choice of a as well. However, the subtlety here is that an optimal a is dependent on the state s, which is why it is denoted $a_{n,s}$. Thus, establishing the convexity of $v^*_n(s)$ would require analyzing how all terms that involve $a_{n,s}$ vary as functions of s.

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

Proof. Rewrite (4.53) as

$$u_n(s) = f(s) + b\Delta u_{n+1}(s+1) + u_{n+1}(s) + \min_{a \in A}\{m(a) - a\Delta u_{n+1}(s)\}. \tag{4.60}$$

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

Fix $s \in \{1, 2, \ldots\}$. Let $a_{n,s}$ be the smallest minimizer of $m(a) - a\Delta v^*_{n+1}(s)$. Let $s' > s$. To show that $a_{n,s'} \geq a_{n,s}$ assume to the contrary that $a_{n,s'} < a_{n,s}$. Then

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

where the first inequality is due to optimality of $a_{n,s'}$ and the second inequality is due to the 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. □

Combining these two lemmas and noting that $a^*_{n,0} = a_1$ gives the main result.

Theorem 4.7. Suppose $f(s)$ and $r_N(s)$ are convex functions of s. Then for $n = 1, \ldots, N-1$, $d^*_n(s)$ is monotone non-decreasing in s.

Note this section established Theorem 4.7 from first principles. Alternatively one could prove it using Theorem 4.5 by choosing V^F to be the set of convex functions on S and showing that V^F is compatible with D^F the set of monotone non-decreasing decision rules.

4.5 State-action value functions

Using the Bellman equation

$$u_n(s) = \max_{a \in A_s} \left\{ \sum_{j \in S} p_n(j|s,a)(r_n(s,a,j) + u_{n+1}(j)) \right\} \tag{4.61}$$

may not be the most effective approach for finding an optimal policy when S is large because computing expectations inside the maximum for each action may be computationally prohibitive. Instead, basing calculations on a recursion for state-action value functions might alleviate this problem by providing a basis for analyzing such models by simulation. They are fundamental in Chapters 10 and 11.

A Bellman-like recursion for state-action value functions requires the concept of a *shifted period*, one that starts immediately after selecting an action at decision epoch n and concludes immediately *after* selecting an action at decision epoch $n+1$. This contrasts with the traditional Markov decision process viewpoint where a period starts immediately before action choice at decision epoch n and ends *prior* to action choice at decision epoch $n+1$.

Figure 4.6 compares these two perspectives. The shifted period perspective represents transitions from *post-decision* states to *post-decision* states while the usual Markov decision process perspective represents transitions from *pre-decision* states to *pre-decision* states.

4.5.1 State-action value function recursion for a fixed policy

The *state-action value function*[21] $q^\pi_n(s,a)$ of the Markovian deterministic[22] policy $\pi = (d_1, \ldots, d_{N-1})$ represents the expected total reward to the end of the planning horizon

[21]This is also commonly referred to as a q-function, following the use of q to represent the function mathematically.

[22]A modification for Markovian randomized policies appears below.

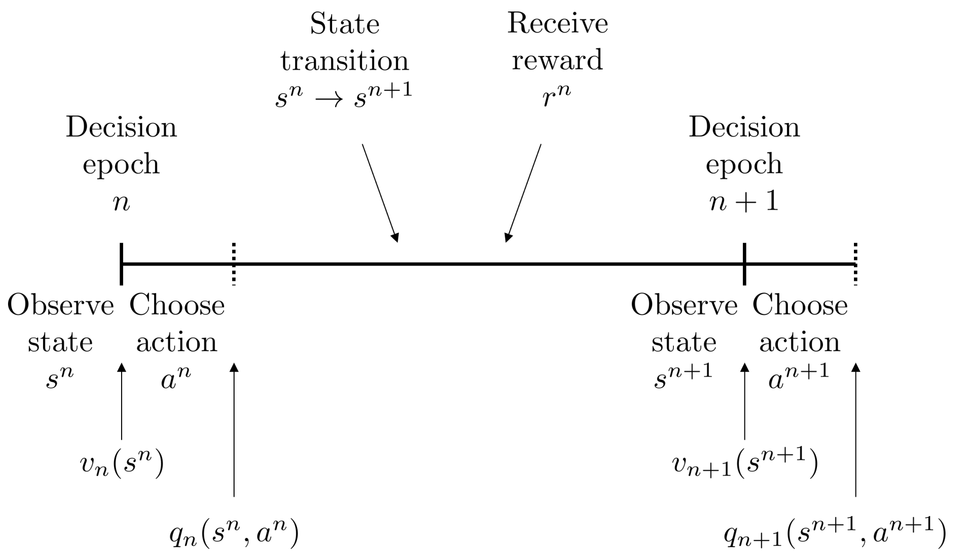


Figure 4.6: Timeline showing that $v_n(s^n)$ is computed after observing state s^n but before choosing action a^n, whereas $q_n(s^n, a^n)$ is computed after observing both s^n and a^n. In this diagram, a period starts at decision epoch n and terminates immediately before decision epoch $n+1$ after observing s^{n+1}.

after choosing action a in state s at decision epoch n and following $(d_{n+1}, d_{n+2}, \ldots, d_{N-1})$ thereafter.

It is easy to show that for $n = 1, 2, \ldots, N-1$, $q_n^\pi(s,a)$ satisfies the following equations.

State-action value function recursion to evaluate a fixed Markovian deterministic policy:

For all $s \in S$ and $a \in A_s$,

$$q_N(s,a) = r_N(s) \tag{4.62}$$

and

$$q_n(s,a) = \sum_{j \in S} p_n(j|s,a)\Big(r_n(s,a,j) + q_{n+1}(j, d_{n+1}(j))\Big) \tag{4.63}$$

for $n = 1, \ldots, N-1$.

In contrast, the value function for π satisfies the following recursions, which appeared as (4.6).

Value function recursion to evaluate a fixed Markovian deterministic policy:

For $s \in S$,

$$u_N(s) = r_N(s) \tag{4.64}$$

and

$$u_n(s) = \sum_{j \in S} p_n(j|s, d_n(s))\big(r_n(s, d_n(s), j) + u_{n+1}(j)\big) \tag{4.65}$$

for $n = 1, 2, \ldots, N-1$.

Some important and subtle distinctions between these two recursions follow:

1. The state-action value function involves d_{n+1} while the value function depends on d_n.
2. The policy enters into (4.63) only through the appearance of $d_{n+1}(j)$ in the expression $q_{n+1}(j, d_{n+1}(j))$. It does not appear in either $p_n(j|s,a)$ or $r_n(s,a,j)$. On the other hand, in (4.65), the policy appears explicitly through $d_n(s)$ in both $p_n(j|s, d_n(s))$ and $r_n(s, d_n(s), j)$ but not explicitly in $u_{n+1}(j)$.
3. Expression (4.65) only depends on a single action $d_n(s)$, whereas (4.63) relates the value of two actions, the action a chosen at decision epoch n and the policy-specific action $d_{n+1}(j)$, where j is the subsequent state to s.
4. When evaluating a deterministic policy, one needs only $q_{n+1}(j, d_{n+1}(j))$ at the next iterate and not $q_{n+1}(j,a)$ for $a \neq d_{n+1}(j)$.
5. If instead π is a Markovian randomized policy, the quantity $q_{n+1}(j, d_{n+1}(j))$ is replaced by $\sum_{a' \in A_s} w^n_{d_n}(a'|j) q_{n+1}(j, a')$.

4.5.2 Revisiting the two-state model from the state-action value function perspective

This section analyzes state-action value functions for the two-state example in Section 4.1.2. Set $N = 3$, the terminal reward $r_3(s_1) = r_3(s_2) = 0$, and evaluate the Markovian deterministic policy $\pi = (d_1, d_2)$, in which

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

and

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

When $n = 2$:

$$q_2^\pi(s_1, a_{1,1}) = p_2(s_1|s_1, a_{1,1})\big(r_2(s_1, a_{1,1}, s_1) + r_3(s_1)\big) + p_2(s_2|s_1, a_{1,1})\big(r_2(s_1, a_{1,1}, s_2) + r_3(s_2)\big)$$

$$
\begin{aligned}
&= 0.8(5+0) + 0.2(-5+0)\\
&= 3
\end{aligned}
$$

and

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

When $n = 1$:

$$
q_1^\pi(s_1, a_{1,2}) = p_1(s_1|s_1, a_{1,2})\big(r_1(s_1, a_{1,2}, s_1) + q_2^\pi(s_1, d_2(s_1))\big) + \\ p_1(s_2|s_1, a_{1,2})\big(r_1(s_1, a_{1,2}, s_2) + q_2^\pi(s_2, d_2(s_2))\big) \quad (4.66)
$$

and

$$
q_1^\pi(s_2, a_{2,2}) = p_1(s_1|s_2, a_{2,2})\big(r_1(s_2, a_{2,2}, s_1) + q_2^\pi(s_1, d_2(s_1))\big) + \\ p_1(s_2|s_2, a_{2,2})\big(r_1(s_2, a_{2,2}, s_2) + q_2^\pi(s_2, d_2(s_2))\big). \quad (4.67)
$$

Substituting values in (4.66) and (4.67) 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-function values for actions that π does not select at decision epochs 1 and 2 because they do not enter into subsequent evaluations and are not required to determine $v_1^\pi(s)$.

4.5.3 Optimal state-action value functions

State-action value functions are more important when seeking optimal policies. 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 follows from an obvious modification of the proof of Theorem 4.2.

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

$$
q_N^*(s, a) = r_N(s) \quad (4.68)
$$

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

$$q^*_{N-1}(s,a) = \sum_{j\in S} p_{N-1}(j|s,a)\big(r_{N-1}(s,a,j) + r_N(j)\big). \tag{4.69}$$

At decision epoch $N-2$, the recursion now must choose the decision at the next decision epoch, $N-1$ optimally so that

$$q^*_{N-2}(s,a) = \max_{a'\in A_s}\left\{\sum_{j\in S} p_{N-1}(j|s,a)\big(r_{N-1}(s,a,j) + q^*_{N-1}(j,a')\big)\right\} \tag{4.70}$$

$$= \sum_{j\in S} p_{N-1}(j|s,a)\big(r_{N-1}(s,a,j) + \max_{a'\in A_s}\{q^*_{N-1}(j,a')\}\big). \tag{4.71}$$

The second equality follows by noting that a' only appears in $q^*_{N-1}(j,a')$ so the maximization can be brought inside the expectation. Continuing in this way it is easy to prove that $q^*_n(s,a)$ satisfies the following recursions.

Bellman equation for state-action value functions:

For $s \in S$ and $a \in A_s$,

$$q_N(s,a) = r_N(s) \tag{4.72}$$

and

$$q_n(s,a) = \sum_{j\in S} p_n(j|s,a)\big(r_n(s,a,j) + \max_{a'\in A_j}\{q_{n+1}(j,a')\}\big) \tag{4.73}$$

for $n = 1,\dots,N-1$.

A few key observations about this development follow:

1. It is important to note that the Bellman equation for $q^*_n(s,a)$ differs from the Bellman equation for $v^*_n(s)$ 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. There is a very close connection between $v^*_n(s)$ and $q^*_n(s,a)$. In particular, for $n = 1,2,\dots,N$

$$v^*_n(s) = \max_{a\in A_s} q^*_n(s,a). \tag{4.74}$$

   It then follows that

$$q^*_n(s,a) = \sum_{j\in S} p_n(j|s,a)(r_n(s,a,j) + v^*_{n+1}(j)). \tag{4.75}$$

   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.*

3. For $n = 1, 2, \dots, N-1$ 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 $v^*_{n+1}(\cdot)$:

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

Using q-functions, and noting equation (4.75), $A^*_{n,s}$ can be written much more succinctly as

$$A^*_{n,s} = \arg\max_{a \in A_s} q^*_n(s,a). \tag{4.76}$$

4. We found that when coding backwards induction based on value functions as in Algorithm 4.3 it was convenient to first evaluate

$$q_n(s,a) = r_n(s,a) + \sum_{j \in S} p_n(j|s,a) u_{n+1}(j)$$

for each $a \in A_s$ and then set

$$u_n(s) = \max_{a \in A_s} q_n(s,a).$$

Backwards induction using q-functions

Algorithm 4.4. Finite horizon policy optimization using state-action value functions

1. **Initialize:** Set $n \leftarrow N-1$, $q_N(s,a) \leftarrow r_N(s)$ for all $s \in S$ and $a \in A_s$, and $d_N(s)$ to be an arbitrary action in A_s.
2. **Iterate:** While $n \geq 1$:
   (a) For all $s \in S$ and $a \in A_s$, evaluate $q_n(s,a)$ according to

   $$q_n(s,a) \leftarrow \sum_{j \in S} p_n(j|s,a)\big(r_n(s,a,j) + q_{n+1}(j, d_{n+1}(j))\big), \tag{4.77}$$

   where $d_{n+1}(j)$ can be defined as an arbitrary action $a \in A_j$. Set

   $$A^*_{n,s} \leftarrow \arg\max_{a \in A_s} q_n(s,a) \tag{4.78}$$

   and choose $d_n(s)$ to be any action in $A^*_{n,s}$.
   (b) $n \leftarrow n - 1$.
3. **Terminate:** Return $q_1(s, d_1(s))$ for all $s \in S$ and $\pi = (d_1, \dots, d_{N-1})$.

It is left 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 4.3).

4.5.4 An example

This section finds an optimal policy in the two-state model using Algorithm 4.4. First set $q_3^*(s,a) = 0$ for all $a \in A_s$ and $s \in S$.

To find q_1^* and q_2^*, requires accounting for each possible action in each state at each decision epoch. For example:

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

Table 4.3 gives all values for $q_n^*(s,a)$ for $n = 1, 2$.

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 4.3: 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 4.3 by noting which actions attain the maximum at each decision epoch. These appear in Table 4.4.

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

Inspecting Table 4.4 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 (4.76) 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 4.2.3 using the corresponding algorithm for value functions.

4.5.5 Optimality equations in the queuing admission control model*

This section considers a finite horizon version of the queuing admission control model in Section 3.4.2. It first compares the traditional Bellman equation for value functions expressed in terms of pre-decision states, with the post-decision state model described in Section 3.4.2. Finally, it provides the Bellman equation using q-functions, which also corresponds to a post-decision state perspective.

Value function optimality equations: Pre-decision state formulation

The optimality equations expressed in terms of value functions for the pre-decision state formulation become, for $j = 0$,

$$u_n\big((0,1)\big) = \max\Big\{R - f(1) + bu_{n+1}\big((1,1)\big) + (1-b)u_{n+1}\big((1,0)\big), \\ -f(0) + bu_{n+1}\big((0,1)\big) + (1-b)u_{n+1}\big((0,0)\big)\Big\} \quad (4.79)$$

and

$$u_n\big((0,0)\big) = -f(0) + bu_{n+1}\big((0,1)\big) + (1-b)u_{n+1}\big((0,1)\big) \quad (4.80)$$

and for $j > 0$,

$$u_n\big((j,1)\big) \\ = \max\Big\{R - f(j+1) + bu_{n+1}\big((j+1,1)\big) + (1-b-w)u_{n+1}\big(j+1,0)\big) + wu_{n+1}\big((j,0)\big), \\ -f(j) + bu_{n+1}\big((j,1)\big) + (1-b-w)u_{n+1}\big((j,0)\big) + wu_{n+1}\big((j-1,0)\big)\Big\} \quad (4.81)$$

and

$$u_n\big(j,0)\big) = -f(j) + bu_{n+1}\big((j,1)\big) + (1-b-w)u_{n+1}\big((j,0)\big) + wu_{n+1}\big((j-1,0)\big) \quad (4.82)$$

for $n = 1,\ldots,N-1$. Observe that (4.80) and (4.82) do not contain the expression "max" since there is no decision when there are no jobs to admit.

Value function optimality equations: Post-decision state formulation

The optimality equations in the alternative post-decision state formulation become, for $j = 0$,

$$u_n(0) \;=\; \max\big\{R - f(1) + bu_{n+1}(1) + (1-b)u_{n+1}(0), -f(0) + u_{n+1}(0)\big\} \quad (4.83)$$

and for $j > 0$,

$$u_n(j) = \max\big\{R - f(j+1) + bu_{n+1}(j+1) + (1-b-w)u_{n+1}(j) + wu_{n+1}(j-1) \\ -f(j) + (1-w)u_{n+1}(j) + wu_{n+1}(j-1)\big\}. \quad (4.84)$$

Note that the main difference between (4.83) and (4.84) is the inclusion of the possibility of a service completion when $j > 0$. Note also that the first term in the "max" corresponds to admitting an arrival if it occurs and the second term to not doing so. Thus the second term does not involve b because whether or not an arrival occurs does not affect the "do not admit" decision.

Clearly the alternative formulation based on the post-decision states provides a more concise recursion.

State-action value function optimality equations

Now consider the optimality equations based on state-action value functions using the post-decision state formulation corresponding to Figure 3.8.

For $j = 0$,

$$q_n(0, a_0) = -f(0) + \max\{q_{n+1}(0, a_0), q_{n+1}(0, a_1)\} \quad (4.85)$$

and

$$q_n(0, a_1) = (1-b)\big(-f(0) + \max\{q_{n+1}(0, a_0), q_{n+1}(0, a_1)\}\big) \\ + b\big(R - f(1) + \max\{q_{n+1}(1, a_0), q_{n+1}(1, a_1)\}\big). \quad (4.86)$$

Since whether or not an arrival occurs doesn't impact the "do not admit" action a_0, when $j > 0$

$$q_n(j, a_0) = (1-w)\big(-f(j) + \max\{q_{n+1}(j, a_0), q_{n+1}(j, a_1)\}\big) \\ + w\big(-f(j-1) + \max\{q_{n+1}(j-1, a_0), q_{n+1}(j-1, a_1)\}\big) \quad (4.87)$$

and for action a_1 and $j > 0$

$$q_n(j, a_1) = (1-b-w)\big(-f(j) + \max\{q_{n+1}(j, a_0), q_{n+1}(j, a_1)\}\big) \\ + w\big(-f(j-1) + \max\{q_{n+1}(j-1, a_0), q_{n+1}(j-1, a_1)\}]\big) \\ + b\big(R - f(j) + \max\{q_{n+1}(j+1, a_0)), q_{n+1}(j+1, a_1)\}\big). \quad (4.88)$$

Some comments on these equations follow:

1. The state-action value function optimality equations involve actions at two decision epochs. For example, in (4.88), a_1 on the left hand side of the equation corresponds to the choice of this action at decision epoch n while the quantity within the "max" corresponds to the action at the decision epoch $n+1$. The expression "max" ensures that the later decision is taken optimality.

2. In the value function optimality equations (4.83) and (4.84) the expectation over the next event is taken *before* evaluating the "max" while in the state-action value function optimality equations (4.85)-(4.88), the expectation over the next event is taken after evaluating the "max". This observation does not affect exact calculation using backwards induction but underlies the use of q-functions in simulations, especially in large models. In this example, simulation means using a random mechanism to generate the next event (arrival, service completion or neither).

Exercise 22 at the end of this chapter provides an opportunity to compare these three approaches to finding an optimal policy in an instance of the admission control model.

4.6 Technical appendix

This appendix shows that the finite horizon policy optimization algorithm (Algorithm 4.3) finds optimal values and optimal policies within the class of Markovian deterministic policies. It then extends this result to establish that these policies are optimal in the class of history-dependent policies.

4.6.1 Finding optimal policies in Π^{MD}

Theorem 4.8. 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. The proof of the first result is 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 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 Theorem 4.1. Hence the result holds for $n = 1, \dots, N$.

The proof of the second result is 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 $t = n+1, \dots, N$. By the definition of d_n^* together with (4.22) and (4.23) it follows for all $s \in S$ that

$$u_n(s) = \max_{a \in A_s} \left\{ \sum_{j \in S} p_n(j|s,a)(r_n(s,a,j) + u_{n+1}(j)) \right\} \tag{4.89}$$

$$= \sum_{j \in S} p_n(j|s, d_n^*(s))(r_n(s,a,d_n^*(s)) + u_{n+1}(j)) \tag{4.90}$$

$$= \sum_{j \in S} p_n(j|s, d_n^*(s))(r_n(s,a,d_n^*(s)) + v_{n+1}^{\pi^*}(j)) \tag{4.91}$$

$$= v_n^{\pi^*}(s), \tag{4.92}$$

where the third equality follows from the induction hypothesis and the last equality follows from Theorem 4.1. Hence the result follows.

□

Corollary 4.2. Let π^* be as defined in Theorem 4.8. Then π^* is optimal in the class of Markov deterministic policies.

The main takeaway from Theorem 4.8 and Corollary 4.2 is that Algorithm 4.3 is guaranteed to find an optimal policy within the class of Markov deterministic policies, along with the corresponding optimal value. However, not only does it find the optimal policy and value starting from the first epoch, it provides a method to construct optimal policies from any decision epoch n to the end of the planning horizon, along with the associated optimal value from epoch n onward.

4.6.2 Finding optimal values in Π^{HR}

Before proving that there exist Markovian deterministic policies that are optimal in the class of history-dependent randomized policies, an algorithm is presented for finding optimal values only. Since results in the chapter Appendix establish that the search for optimal policies can be restricted to Markovian deterministic policies, this algorithm is not intended for use in practice.

Algorithm 4.5. Finite horizon policy optimization in Π^{HR}

1. **Initialize:** Set $n \leftarrow N-1$ and $u_N'(h^N) \leftarrow r_N(s^N)$ for all $h^N = (h^{N-1}, a^{N-1}, s^N) \in H_N$.

2. **Iterate:** While $n \geq 1$:
   (a) For all $h^n = (h^{n-1}, a^{n-1}, s^n) \in H_n$, evaluate $u'_n(h^n)$ according to

$$u'_n(h^n) \leftarrow \sup_{w \in \mathscr{P}(A_{s^n})} \left\{ \sum_{a \in A_{s^n}} w(a) \sum_{j \in S} p_n(j|s^n, a)\big(r_n(s^n, a, j) + u'_{n+1}((h^n, a, j))\big) \right\}. \tag{4.93}$$

   (b) $n \leftarrow n - 1$.
3. **Terminate:** Return $u'_1(s)$ for all $s \in S$ and $\pi = (d_1, \ldots, d_{N-1})$.

Equation (4.93) provides the most general form of the Bellman equation. A similar but slightly more tedious proof than that of Theorem 4.8 establishes that this algorithm finds optimal values. The following proposition states this formally; proving it is left as an exercise.

Proposition 4.1. Suppose for each $h^n \in H_n$, $n = 1, \ldots, N$, $u'_n(h^n)$ is determined by Algorithm 4.5. Then $u'_n(h^n) = v^*_n(h^n)$ and $u'_1(s) = v^*(s)$ for all $s \in S$.

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

What remains to be shown is that in (4.93), 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. This is accomplished through two propositions. The first is an immediate consequence of the second part of Lemma 2.1 and its proof is omitted.

Proposition 4.2. 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

$$\begin{aligned} u'_n(h^n) &= \sup_{w \in \mathscr{P}(A_{s^n})} \left\{ \sum_{a \in A_{s^n}} w(a) \sum_{j \in S} p_n(j|s^n, a)(r_n(s^n, a, j) + u'_{n+1}(h^{n+1})) \right\} \\ &= \max_{a \in A_{s^n}} \left\{ \sum_{j \in S} p_n(j|s^n, a)(r_n(s^n, a, j) + u'_{n+1}(h^{n+1})) \right\}. \end{aligned} \tag{4.94}$$

Proposition 4.3. 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 $n = 1, \ldots, N-1$ 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, $u'_n(h^n) = u_n(s^n)$.

Proof. The proof is 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 $t = n+1, \ldots, N$. By the induction hypothesis and (4.94),

$$\begin{aligned} u'_n(h^n) &= \max_{a \in A_{s^n}} \left\{ \sum_{j \in S} p_n(j|s^n, a)(r_n(s^n, a, j) + u'_{n+1}(h^{n+1})) \right\} \\ &= \max_{a \in A_{s^n}} \left\{ \sum_{j \in S} p_n(j|s^n, a)(r_n(s^n, a, j) + u_{n+1}(j)) \right\} \end{aligned} \tag{4.95}$$

where $h^n = (h^{n-1}, a^n, s^n)$. Observe that all the quantities on the right hand side of (4.95) 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 4.2 is that randomization is not necessary when there are a finite number of actions. Proposition 4.3 states that only the current state matters if the transition probabilities, reward functions, and especially the terminal reward do not depend on states prior to the current state. Together, these two results establish the following equalities:

$$\sup_{\pi \in \Pi^{\mathrm{HR}}} v_n^\pi(h^n) = \max_{\pi \in \Pi^{\mathrm{HD}}} v_n^\pi(h^n) = \max_{\pi \in \Pi^{\mathrm{MD}}} v_n^\pi(s^n). \tag{4.96}$$

Thus, combining Propositions 4.1, 4.2 and 4.3 provides a proof of Theorem 4.2. As a consequence of Theorem 4.2, 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 $s \in S$ and

$$v^*(s) \geq v^\pi(s)$$

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

Bibliographic remarks

The use of the backwards 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 (see Gessford and Karlin [1958]) and Wald's research on sequential analysis [Wald, 1947]. See Chapter 4 in Puterman [1994] for a more thorough discussion of 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].

The proof of optimality of monotone policies in queuing service rate control follows Lippman [1975]. That result can also be proved using submodularity, see Section 4.4.2 of Puterman [1994]. The use of submodularity in Markov decision processes originates with the elegant paper of Serfozo [1976]. As noted in Chapter 3, the online dating problem originates with Cayley [1875]. The analysis herein is based on Bather [1982].

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

Exercises

1. Consider the variant of the two-state model analyzed in Section 4.1.2. Propose a randomized policy and evaluate it using backwards induction and sample path enumeration.

2. Find an optimal policy for the model in Example 2.2 for $N = 2$ by enumerating and evaluating all Markovian deterministic policies using Algorithm 4.1.

3. Find an optimal policy for the model in Example 2.2 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.

4. Prove 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 $n = 1, \ldots, N$.

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

(a) Represent the model graphically as in Figure 2.7.

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

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

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

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

7. 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 optimality equations in terms of q-functions.

8. For Example 2.2 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 $q_{2,6} = 0.6$

9. Consider the queuing service rate control model in Section 4.3.1. In that example, the terminal cost was $r_N(s) = 0$ in all states so that the optimal policy was to use the least costly service rate in all states at the last decision epoch. To further investigate the impact of the terminal cost, repeat the analysis with linear, quadratic and cubic terminal costs. Discuss how the terminal cost impacts optimal policies.

10. Solve the queuing service rate control problem from Section 4.3.1 with the following changes to the parameters: a horizon length of $N = 10$, a delay cost of $f(s) = \sqrt{s}$ and a serving cost of $m(a) = 5a^3$. Comment on the structure of the optimal policy compared to the policies depicted in that section.

11. Solve the restaurant revenue management problem (Exercise 15 from Chapter 3) over a two-hour planning horizon. Clearly state and graphically represent the optimal policy.

12. Consider a 5-period version of the periodic inventory review problem of Section 3.1 with a finite warehouse 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.

13. Consider the lion hunting problem from Section 3.5 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.

14. Consider the network shown in Figure 4.7.

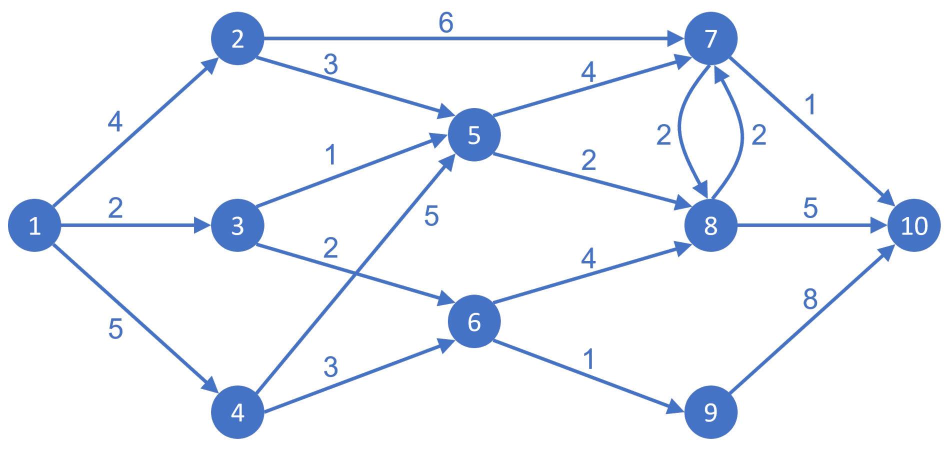


Figure 4.7: Network for Problem 14.

(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 backwards induction. Note that the use of periods in this example is artificial.

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

(c) Find the longest path from node 1 to node 10 using backwards 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.

15. 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.

16. Solve the single machine maintenance problem (Exercise 12 in Chapter 3), assuming it is a finite horizon problem with $N = 5$, using the following data: $r = 100$, $p(i) = (1 - e^{-i})/(1 + e^{-i})$, $c_A = 60$, $c_B = 20$.

17. 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, assume the best offer follows a similar discrete uniform distribution, but centered at the best offer from the previous day.

18. Consider the house selling problem in a hot housing market. The problem parameters are the same as in Exercise 17, 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.

19. 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 17.

20. Formulate and solve a variant of the online dating model of Section 4.3.3 in which the objective is to maximize the probability of selecting one of the two best matches.

21. Consider the queuing admission control problem from Section 4.5.5. Derive the optimality equations for the value and state-action 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 $N = 10$. Assume $b = 0.2$, $w = 0.2$, $f(j) = 5j$, and $R = 30$. At the final epoch, assume any remaining jobs in the system are penalized at a cost of 10 per job.

    (a) Solve this problem using both the value function optimality equations and the state-action value function optimality equations.

    (b) Compare and discuss the computational efforts in each approach.

23. Consider the machine maintenance model in Section 4.4.2.

    (a) Verify the result that a control-limit policy is optimal by numerically solving the problem over a range of parameter values of your choosing. It is easiest to consider instances in which the holding costs and transition probabilities are stationary.

    (b) *Provide conditions on $f_n(s)$ and p_n that ensure that $0 \leq s_n \leq M - 1$ for all n where s_n is defined in (4.52).

    (c) *Provide conditions on the model components that ensure that the control limit is monotone in the decision epoch n.

24. Express the state-action value function recursion that replaces (4.63) when evaluating a fixed Markovian *randomized* policy.

25. Prove Theorem 4.1 for Markovian randomized policies and history-dependent randomized policies.