SUPPLEMENT 1 TO CHAPTER 19

A Policy Improvement Algorithm for Finding Optimal Policies

Chapter 19 described two methods for deriving an optimal policy for a Markov decision process: *exhaustive enumeration* and *linear programming*. Exhaustive enumeration is useful because it is both quick and straightforward for very small problems. Linear programming can be used to solve vastly larger problems, and software packages for the simplex method are very widely available.

We now present a third popular method, namely, a *policy improvement algorithm.* The key advantage of this method is that it tends to be very efficient because it usually reaches an optimal policy in a relatively small number of iterations (far fewer than for the simplex method with a linear programming formulation).

Consider the model for Markov decision processes presented in Sec. 19.2. As a joint result of the current state i of the system and the decision $d_i(R) = k$ when operating under policy R, two things occur. First, an (expected) cost C_{ik} is incurred that depends upon only the observed state of the system and the decision made. Second, the system moves to state j at the next observed time period, with transition probability given by $p_{ij}(k)$. If, in fact, state j influences the cost that has been incurred, then C_{ik} is calculated as follows. Let

$q_{ij}(k)$ = expected cost incurred when the system is in state i, decision k is made, and the system evolves to state j at the next observed time period.

Then

$$C_{ik} = \sum_{j=0}^{M} q_{ij}(k)p_{ij}(k).$$

Preliminaries

Referring to the description and notation for Markov decision processes given at the beginning of Sec. 19.2, we can show that, for any given policy R, there exist values $g(R)$, $v_0(R)$, $v_1(R)$, . . . , $v_M(R)$ that satisfy

$$g(R) + v_i(R) = C_{ik} + \sum_{j=0}^{M} p_{ij}(k)v_j(R), \quad \text{for } i = 0, 1, 2, \ldots, M.$$

We now shall give a heuristic justification of these relationships and an interpretation for these values.

Denote by $v_i^n(R)$ the total expected cost of a system starting in state i (beginning the first observed time period) and evolving for n time periods. Then $v_i^n(R)$ has two components: C_{ik}, the cost incurred during the first observed time period, and $\sum_{j=0}^{M} p_{ij}(k)\, v_j^{n-1}(R)$, the total expected cost of the system evolving over the remaining $n - 1$ time periods. This gives the *recursive equation*

$$v_i^n(R) = C_{ik} + \sum_{j=0}^{M} p_{ij}(k)\, v_j^{n-1}(R), \quad \text{for } i = 0, 1, 2, \ldots, M,$$

where $v_i^1(R) = C_{ik}$ for all i.

It will be useful to explore the behavior of $v_i^n(R)$ as n grows large. Recall that the (long-run) expected average cost per unit time following any policy R can be expressed as

$$g(R) = \sum_{i=0}^{\mathrm{M}} \pi_i C_{ik},$$

which is independent of the starting state i. Hence, $v_i^n(R)$ behaves approximately as $n\, g(R)$ for large n. In fact, if we neglect small fluctuations, $v_i^n(R)$ can be expressed as the sum of two components

$$v_i^n(R) \approx n\, g(R) + v_i(R),$$

where the first component is independent of the initial state and the second is dependent upon the initial state. Thus, $v_i(R)$ can be interpreted as the effect on the total expected cost due to starting in state i. Consequently,

$$v_i^n(R) - v_j^n(R) \approx v_i(R) - v_j(R),$$

so that $v_i(R) - v_j(R)$ is a measure of the effect of starting in state i rather than state j.

Letting n grow large, we now can substitute $v_i^n(R) = n\, g(R) + v_i(R)$ and $v_j^{n-1}(R) = (n - 1)g(R) + v_j(R)$ into the *recursive equation.* This leads to the system of equations given in the opening paragraph of this subsection.

Note that this system has $M + 1$ equations with $M + 2$ unknowns, so that one of these variables may be chosen arbitrarily. By convention, $v_M(R)$ will be chosen equal to zero. Therefore, by solving the system of linear equations, we can obtain $g(R)$, the (long-run) expected average cost per unit time when policy R is followed. In principle, all policies can be enumerated and that policy which minimizes $g(R)$ can be found. However, even for a moderate number of states and decisions, this technique is cumbersome. Fortunately, there exists an algorithm that can be used to evaluate policies and find the optimal one without complete enumeration, as described next.

The Policy Improvement Algorithm[1]

The algorithm begins by choosing an arbitrary policy R_1. It then solves the system of equations to find the values of $g(R_1)$, $v_0(R)$, $v_1(R)$, . . . , $v_{M-1}(R)$ [with $v_M(R) = 0$]. This

[1]This algorithm assumes that the Markov chain associated with the transition matrices used by the Markov decision process is irreducible, i.e., any state can be reached eventually from any other state.

step is called *value determination.* A better policy, denoted by R_2, is then constructed. This step is called *policy improvement.* These two steps constitute an iteration of the algorithm. Using the new policy R_2, we perform another iteration. These iterations continue until two successive iterations lead to identical policies, which signifies that the optimal policy has been obtained. The details are outlined below.

Summary of the Policy Improvement Algorithm

Initialization: Choose an arbitrary initial trial policy R_1. Set $n = 1$.
Iteration n:
Step 1: Value determination: For policy R_n, use $p_{ij}(k)$, C_{ik}, and $v_M(R_n) = 0$ to solve the system of $M + 1$ equations

$$g(R_n) = C_{ik} + \sum_{j=0}^{M} p_{ij}(k)\, v_j(R_n) - v_i(R_n), \quad \text{for } i = 0, 1, \ldots, M.$$

for all $M + 1$ unknown values of $g(R_n)$, $v_0(R_n)$, $v_1(R_n)$, . . . , $v_{M-}(R_n)$.
Step 2: Policy improvement: Using the current values of $v_i(R_n)$ computed for policy R_n, find the alternative policy R_{n+1} such that, for each state i, $d_i(R_{n+1}) = k$ is the decision that minimizes

$$C_{ik} + \sum_{j=0}^{M} p_{ij}(k)\, v_j(R_n) - v_j(R_n),$$

i.e., for each state i,

$$\underset{k=1,\,2,\,\ldots,K}{\text{Minimize}} \left[C_{ik} + \sum_{j=0}^{M} p_{ij}(k)\, v_j(R_n) - v_i(R_n)\right],$$

and then set $d_i(R_{n+1})$ equal to the minimizing value of k. This procedure defines a new policy R_{n+1}.
Optimality test: The current policy R_{n+1} is optimal if this policy is identical to policy R_n. If it is, stop. Otherwise, reset $n = n + 1$ and perform another iteration.

Two key properties of this algorithm are

1. $g(R_{n+1}) \leq (R_n)$, for $n = 1, 2, \ldots$
2. The algorithm terminates with an optimal policy in a finite number of iterations.[2]

Solving the Prototype Example by the Policy Improvement Algorithm

Referring to the prototype example presented in Sec. 19.1, we outline the application of the algorithm next.

Initialization. For the initial trial policy R_1, we arbitrarily choose the policy that calls for replacement of the machine (decision 3) when it is found to be in state 3, but doing nothing (decision 1) in other states. This policy, its transition matrix, and its costs are summarized next.

[2]This termination is guaranteed under the assumptions of the model given in Sec. 19.2, including particularly the (implicit) assumptions of a finite number of states ($M + 1$) and a finite number of decisions *(K),* but not necessarily for more general models. See R. Howard, *Dynamic Programming and Markov Processes,* M.I.T. Press, Cambridge, MA, 1960. Also see pp. 1291–1293 in A. F. Veinott, Jr., "On Finding Optimal Policies in Discrete Dynamic Programming with No Discounting," *Annals of Mathematical Statistics,* **37:** 1284–1294, 1966.

Policy R_1

State	Decision
0	1
1	1
2	1
3	3

Transition matrix

State	0	1	2	3
0	0	$\frac{7}{8}$	$\frac{1}{16}$	$\frac{1}{16}$
1	0	$\frac{3}{4}$	$\frac{1}{8}$	$\frac{1}{8}$
2	0	0	$\frac{1}{2}$	$\frac{1}{2}$
3	1	0	0	0

Costs

State	C_{ik}
0	0
1	1,000
2	3,000
3	6,000

Iteration 1. With this policy, the value determination step requires solving the following four equations simultaneously for $g(R_1)$, $v_0(R_1)$, $v_1(R_1)$, and $v_2(R_1)$ [with $v_3(R_1) = 0$].

$$g(R_1) = \quad + \frac{7}{8}v_1(R_1) + \frac{1}{16}v_2(R_1) - v_0(R_1).$$
$$g(R_1) = 1{,}000 + \frac{3}{4}v_1(R_1) + \frac{1}{8}v_2(R_1) - v_1(R_1).$$
$$g(R_1) = 3{,}000 + \frac{1}{2}v_2(R_1) - v_2(R_1).$$
$$g(R_1) = 6{,}000 + v_0(R_1).$$

The simultaneous solution is

$$g(R_1) = \frac{25{,}000}{13} = 1{,}923$$
$$v_0(R_1) = -\frac{53{,}000}{13} = -4{,}077$$
$$v_1(R_1) = -\frac{34{,}000}{13} = -2{,}615$$
$$v_2(R_1) = \frac{28{,}000}{13} = 2{,}154.$$

Step 2 (policy improvement) can now be applied. We want to find an improved policy R_2 such that decision k in state i minimizes the corresponding expression below.

State 0: $C_{0k} - p_{00}(k)(4{,}077) - p_{01}(k)(2{,}615) + p_{02}(k)(2{,}154) + 4{,}077$
State 1: $C_{1k} - p_{10}(k)(4{,}077) - p_{11}(k)(2{,}615) + p_{12}(k)(2{,}154) + 2{,}615$
State 2: $C_{2k} - p_{20}(k)(4{,}077) - p_{21}(k)(2{,}615) + p_{22}(k)(2{,}154) - 2{,}154$
State 3: $C_{3k} - p_{30}(k)(4{,}077) - p_{31}(k)(2{,}615) + p_{32}(k)(2{,}154)$.

Actually, in state 0, the only decision allowed is decision 1 (do nothing), so no calculations are needed. Similarly, we know that decision 3 (replace) must be made in state 3. Thus, only states 1 and 2 require calculation of the values of these expressions for alternative decisions.

For state 1, the possible decisions are 1 and 3. For each one, we show below the corresponding C_{1k}, the $p_{1j}(k)$, and the resulting value of the expression:

	State 1						
Decision	C_{1k}	$p_{10}(k)$	$p_{11}(k)$	$p_{12}(k)$	$p_{13}(k)$	Value of Expression	
1	1,000	0	$\frac{3}{4}$	$\frac{1}{8}$	$\frac{1}{8}$	1,923	← Minimum
3	6,000	1	0	0	0	4,538	

Since decision 1 minimizes the expression, it is chosen as the decision to be made in state 1 for policy R_2 (just as for policy R_1).

The corresponding results for state 2 are shown below for its three possible decisions.

	State 2						
Decision	**C_{2k}**	**$p_{20}(k)$**	**$p_{21}(k)$**	**$p_{22}(k)$**	**$p_{23}(k)$**	**Value of Expression**	
1	3,000	0	0	$\frac{1}{2}$	$\frac{1}{2}$	1,923	
2	4,000	0	1	0	0	−769	← Minimum
3	6,000	1	0	0	0	−231	

Therefore, decision 2 is chosen as the decision to be made in state 2 for policy R_2. Note that this is a change from policy R_1.

We summarize our new policy, its transition matrix, and its costs below:

Policy R_2

State	Decision
0	1
1	1
2	2
3	3

Transition matrix

State	0	1	2	3
0	0	$\frac{7}{8}$	$\frac{1}{16}$	$\frac{1}{16}$
1	0	$\frac{3}{4}$	$\frac{1}{8}$	$\frac{1}{8}$
2	0	1	0	0
3	1	0	0	0

Costs

State	C_{ik}
0	0
1	1,000
2	4,000
3	6,000

Since this policy is not identical to policy R_1, the optimality test says to perform another iteration.

Iteration 2. For step 1 (value determination), the equations to be solved for this policy are shown below:

$$g(R_2) = \quad + \frac{7}{8}v_1(R_2) + \frac{1}{16}v_2(R_2) - v_0(R_2).$$

$$g(R_2) = 1{,}000 + \frac{3}{4}v_1(R_2) + \frac{1}{8}v_2(R_2) - v_1(R_2).$$

$$g(R_2) = 4{,}000 + v_1(R_2) - v_2(R_2).$$

$$g(R_2) = 6{,}000 + v_0(R_2).$$

The simultaneous solution is

$$g(R_2) = \frac{5{,}000}{3} = 1{,}667$$

$$v_0(R_2) = -\frac{13{,}000}{3} = -4{,}333$$

$$v_1(R_2) = -3{,}000$$

$$v_2(R_2) = -\frac{2{,}000}{3} = -667.$$

Step 2 (policy improvement) can now be applied. For the two states with more than one possible decision, the expressions to be minimized are

State 1: $C_{1k} - p_{10}(k)(4{,}333) - p_{11}(k)(3{,}000) - p_{12}(k)(667) + 3{,}000$
State 2: $C_{2k} - p_{20}(k)(4{,}333) - p_{21}(k)(3{,}000) - p_{22}(k)(667) + 667.$

The first iteration provides the necessary data (the transition probabilities and C_{ik}) required for determining the new policy, except for the values of each of these expressions for each of the possible decisions. These values are

Decision	Value for State 1	Value for State 2
1	1,667	3,333
2	—	1,667
3	4,667	2,334

Since decision 1 minimizes the expression for state 1 and decision 2 minimizes the expression for state 2, our next trial policy R_3 is

Policy R_3

State	Decision
0	1
1	1
2	2
3	3

Note that policy R_3 is identical to policy R_2. Therefore, the optimality test indicates that this policy is optimal, so the algorithm is finished.

Another example illustrating the application of this algorithm is included in your OR Tutor. The Solved Examples section for Chapter 19 on the book's website provides **an additional example** as well. The IOR Tutorial also includes an *interactive* procedure for efficiently learning and applying the algorithm.

LEARNING AIDS FOR THIS SUPPLEMENT ON THIS WEBSITE

A Solved Example:

Examples for Chapter 19

A Demonstration Example in OR Tutor:

Policy Improvement Algorithm—Average Cost Case

Interactive Procedures in IOR Tutorial:

Enter Markov Decision Model
Interactive Policy Improvement Algorithm—Average Cost

Glossary for Chapter 19

See Appendix 1 for documentation of the software.

PROBLEMS

The symbols to the left of some of the problems (or their parts) have the following meaning:

D: The demonstration example listed above may be helpful.
I: We suggest that you use the corresponding interactive procedure listed above (the printout records your work).

D,I **19S1-1.** Use the policy improvement algorithm to find an optimal policy for Prob. 19.2-2.

D,I **19S1-2.** Use the policy improvement algorithm to find an optimal policy for Prob. 19.2-3.

D,I **19S1-3.** Use the policy improvement algorithm to find an optimal policy for Prob. 19.2-4.

D,I **19S1-4.** Use the policy improvement algorithm to find an optimal policy for Prob. 19.2-5.

D,I **19S1-5.** Use the policy improvement algorithm to find an optimal policy for Prob. 19.2-6.

D,I **19S1-6.** Use the policy improvement algorithm to find an optimal policy for Prob. 19.2-7.

D,I **19S1-7.** Use the policy improvement algorithm to find an optimal policy for Prob. 19.2-8.

D,I **19S1-8.** Consider the blood-inventory problem presented in Prob. 28.5-5 (see Chap. 28 on this website). Suppose now that the number of pints of blood delivered (on a regular delivery) can be specified at the time of delivery (instead of using the old policy of receiving 1 pint at each delivery). Thus, the number of pints delivered can be 0, 1, 2, or 3 (more than 3 pints can never be used). The cost of regular delivery is $50 per pint, while the cost of an emergency delivery is $100 per pint. Starting with the policy of taking one pint at each regular delivery if the number of pints on hand just prior to the delivery is 0, 1, or 2 pints (so there never is more than 3 pints on hand), perform two iterations of the policy improvement algorithm. (Because so few pints are kept on hand and the oldest pint always is used first, you now can ignore the remote possibility that any pints will reach 21 days on the shelf and need to be discarded.)