MATH 3817: Dynamic Programming and Stochastic Programming **Fall 2018**

Handout 4: Probabilistic Dynamic Programming Problems

Instructor: Zirui Zhou October 11, 2018

1 Probabilistic Dynamic Programming Problems

Recall from our study of deterministic dynamic programming that many recursions were of the following form:

$$f_t^*(\text{current state}) = \min_{\text{all feasible decisions}} \{ \text{ costs during current stage} + f_{t+1}^*(\text{new state})\}.$$

In deterministic dynamic programming problems, both the *costs during current stage* and *the state in the next stage* are determined with certainty by the current state and current decision. However, in many pratical problems, these factors may not be known with certainty, even if the current state and decision are known.

- Consider the employment level model in Section 1.2 of Handout 3. In that example, we assumed that we know the minimum labors required for each time period (week/season/year). However, in the real world, these minimum labors are usually inferred from historical data. Thus a more reasonable assumption would be that we know the distributions of minimum labors in each time period. For example, instead of saying that the minimum labor for operating a shop in Spring is 255, we may assume that the minimum labor in Spring follows the distribution in Table 1. Moreover, instead of requiring the employment level to be strictly

Minimum Labor in Spring	Probability
245	0.2
250	0.3
255	0.5

Table 1: Minimum labor in Spring

  no less than the minimum labor, we may assume that if a minimum labor is not satisfied, a penalty will be incurred. In such a case, the cost during each stage is no longer deterministic. For example, if we make a decision to employ 250 workers in spring, we may incur (i) maintenance cost with probability 0.2, (ii) no cost with probability 0.3, (iii) a penalty cost with probability 0.5.

- Consider the investment model in Section 1.4 of Handout 4. In that example, we assumed that the bonus rates provided by the two banks are fixed. However, in the real world, these bonus rates usually depend on the total profit of banks in that year. Thus a more realistic assumption would be that the bonus rate in year i follows from a distribution. For example, instead of saying that the bonus rates of Bank A and Bank B in year 1 are 1.8% and 2.3% respectively, we may assume the bonus rates follows the distribution in Table 2. In such a

Bonus Rates for Year 1		
Bank A	Bank B	Probability
1.6%	2.0%	0.3
1.8%	2.3%	0.5
2.0%	2.6%	0.2

Table 2: Possible bonus rates of Bank A and Bank B for year 1

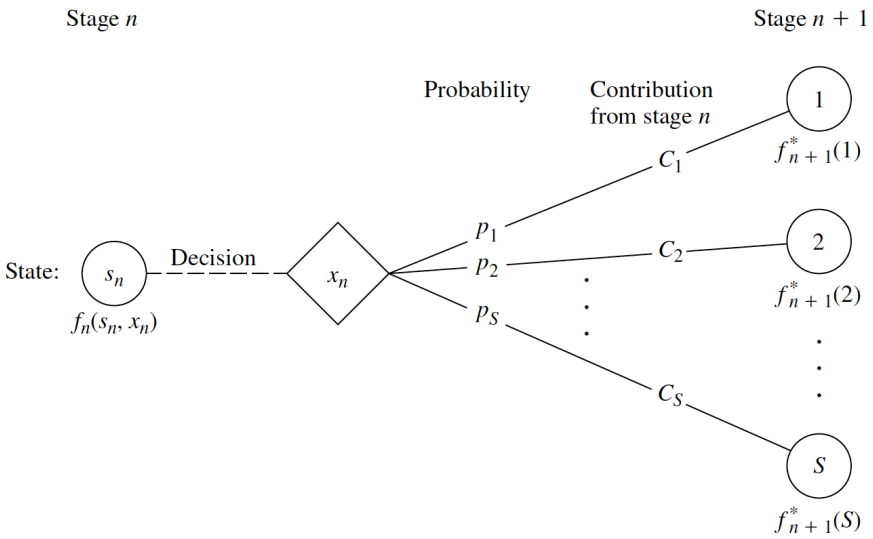

Figure 1: The basic structure for probabilistic dynamic programming

case, the cost/reward in each stage is deterministic, because once you decide to put x_1 into Bank A and $s_1 - x_1$ into Bank B, the value of this investment at the end of year n will be $(1 + r_a)^n x_1 + (1 + r_b)^n (s_1 - x_1)$. This is the reward in stage 1 determined by state s_1 and decision x_1, and it is determined with certainty. However, the transition from s_1 to s_2 is no longer deterministic. In particular, from Table 2, we have

$$\begin{aligned} s_2 &= P_2 + 0.016x_1 + 0.020(s_1 - x_1), \quad \text{with probability } 0.3, \\ s_2 &= P_2 + 0.018x_1 + 0.023(s_1 - x_1), \quad \text{with probability } 0.5, \\ s_2 &= P_2 + 0.020x_1 + 0.026(s_1 - x_1), \quad \text{with probability } 0.2. \end{aligned}$$

The above two examples demonstrate that in the many dynamic programming problems, both the costs during current stage and the state in the next stage can be determined with uncertainty. Dynamic programming problems of this type are usually referred to as probabilistic dynamic programming (PDP). The basic structure for probabilistic dynamic programming is described diagrammatically in Figure 1. For the purposes of this diagram, we let S denote the number of possible states at stage $n+1$ and label these states on the right side as $1, 2, \ldots, S$. The system goes to state i with probability p_i $(i = 1, 2, \ldots, S)$ given state s_n and decision x_n at stage n. If the system goes

to state i, C_i is the contribution of stage n to the objective function.

When Figure 1 is expanded to include all the possible states and decisions at all the stages, it is sometimes referred to as a **decision tree**. If the decision tree is not too large, it provides a useful way of summarizing the various possibilities.

Because of the probabilistic structure, the relationship between $f_n(s_n, x_n)$ and the $f^*_{n+1}(s_{n+1})$ necessarily is somewhat more complicated than that for deterministic dynamic programming. The precise form of this relationship will depend upon the form of the overall objective function. To illustrate, suppose that the objective is to *minimize* the *expected sum* of the cost from the individual stages. In this case, $f_n(s_n, x_n)$ represents the minimum expected sum from stage n onward, given that the state and decision and stage n are s_n and x_n, respectively. Consequently,

$$f_n(s_n, x_n) = \sum_{s_{n+1}=1}^{S} p_i[C_i + f^*_{n+1}(s_{n+1})],$$

with

$$f^*_n(s_n) \equiv \min_{x_n} f_n(s_n, x_n).$$

In this handout, we will present several applications that can be formulated as a probabilistic dynamic programming problem. When study these applications, you are encouraged to pay special attention to the definition of states and the probabilistic transitions between stages.

1.1 Milk Distribution Problem

Example: For a price of \$1/gallon, the Safeco Supermarket chain has purchased 6 gallons of milk from a local dairy. Each gallon of milk is sold in the chain's three stores for \$2/gallon. The dairy must buy back for \$0.5/gallon any milk that is left at the end of the day. Unfortunately for Safeco, demand for each of the chain's three stores is uncertain. Past data indicate that the daily demand at each store is as shown in Table 3. Safeco wants to allocate the 6 gallons of milk to the three stores so as to maximize the expected net daily profit (revenues minus costs) earned from milk. Use dynamic programming to determine how Safeco should allocate the 6 gallons of milk among the three stores. ⊠

Solution. With the exception of the fact that the demand (and therefore the revenue) is uncertain, this problem is very similar to the resource allocation problems studied in Handout 2. Observe that since Safeco's daily purchase costs are always \$6, we may concentrate our attention on the problem of allocating the milk to maximize daily expected revenue earned from the 6 gallons.

The elements of probabilistic dynamic programming are:

1. Stage i: to allocate milk to store i;
2. Decision x_i: the amount of milk allocated to store i;
3. State s_i: the amount of milk still left for allocation to stores $i, i+1, \ldots 3$;
4. $f_i(s_i, x_i)$: the maximum expected revenue eared from the stores $i, i+1, \ldots, 3$, given that there are s_i gallons of milk for allocation to stores $i, i+1, \ldots, 3$ and we decide to allocate x_i gallons of milk to store i;

	Daily Demand (gallons)	Probability
Store 1	1	0.6
	2	0
	3	0.4
Store 2	1	0.5
	2	0.1
	3	0.4
Store 3	1	0.4
	2	0.3
	3	0.3

Table 3: Probability Distributions for Daily Milk Demand

5. $f_i^*(s_i)$: the maximum expected revenue eared from the stores $i, i+1, \ldots, 3$, given that there are s_i gallons of milk for allocation to stores $i, i+1, \ldots, 3$;

6. $x_i^*(s_i)$: the optimal decision at stage i that gives the maximum expected revenue $f_i^*(s_i)$, i.e.,

$$f_i^*(s_i) = f_i(s_i, x_i^*(s_i))$$

Given s_i and x_i, the state in the next stage is determined with certainty, as $s_{i+1} = s_i - x_i$ by definition. However, the reward earned in stage i (the revenue earned from store i) is random. For example, if we allocate 3 gallons to store 1, then the revenue earned from store 1 is

$$\begin{aligned} &2 \times 1 + 0.5 \times (3-1) = 3, && \text{if the daily demand is 1 (with probability 0.6)} \\ &2 \times 3 = 6, && \text{if the daily demand is 3 (with probability 0.4)} \end{aligned}$$

Thus, the expected revenue eared from store 1 if we allocate 3 gallons to store 1 is

$$0.6 \times 3 + 0.4 \times 6 = 4.2.$$

In what follows, we will denote by $r_i(x_i)$ the expected revenue earned from x_i gallons assigned to store i. We have just derived that $r_1(3) = 4.2$. By definition, we have

$$f_i(s_i, x_i) = r_i(x_i) + f_{i+1}^*(s_i - x_i),$$

and thus the following recursion

$$f_i^*(s_i) = \max_{x_i = 0, \ldots, s_i} \{r_i(x_i) + f_{i+1}^*(s_i - x_i)\}. \tag{1}$$

Next we use the working-backward procedure to solve the problem.

Step 1: $i = 3$. Notice that the demand for each store is no more than 3 gallons. Thus the possible states s_3 are $0, 1, 2, 3$, because otherwise ($s_3 \geq 4$) we can always allocate $s_3 - 3$ gallons to store 1 or 2 and the total expected revenue will increase. By definition $f_3^*(s_3)$ is the maximum

expected revenue earned from store 3 given s_i gallons of milk is available for allocation to store 3. Recall that $r_3(x_3)$ is the expected revenue from store 3 if x_3 gallons of milk is allocated to store 3. Thus, we have $f_3^*(s_3) = \max\{r_3(x_3) : x_3 = 0, 1, \ldots, s_3\}$. By a routine computation, we have

$$r_3(0) = 0, \quad r_3(1) = 2, \quad r_3(2) = 3.4, \quad r_3(3) = 4.35.$$

Thus

$$\begin{aligned}
&f_3^*(0) = \max\{r_3(x_3) : x_3 = 0\} = r_3(0) = 0, && x_3^*(0) = 0;\\
&f_3^*(1) = \max\{r_3(x_3) : x_3 = 0, 1\} = r_3(1) = 2, && x_3^*(1) = 1;\\
&f_3^*(2) = \max\{r_3(x_3) : x_3 = 0, 1, 2\} = r_3(2) = 3.4, && x_3^*(1) = 2;\\
&f_3^*(3) = \max\{r_3(x_3) : x_3 = 0, 1, 2, 3\} = r_3(1) = 4.35, && x_3^*(1) = 3.
\end{aligned}$$

Step 2: $i = 2$. The possible states s_2 are $0, 1, 2, 3, 4, 5, 6$, and the decision x_2 is in the range $0, 1, \ldots, \min\{s_2, 3\}$, because we will never allocate more than 3 gallons of milk to store 2. By a routine computation, we have

$$r_2(0) = 0, \quad r_2(1) = 2, \quad r_2(2) = 3.25, \quad r_2(3) = 4.35.$$

Thus, using the recursion (1)

$$\begin{aligned}
&f_2^*(0) = r_2(0) + f_3^*(0) = 0 && x_2^*(0) = 0;\\
&f_2^*(1) = \max\begin{cases} r_2(0) + f_3^*(1) = 2 \\ r_2(1) + f_3^*(0) = 2 \end{cases} && x_2^*(1) = 0 \text{ or } 1;\\
&f_2^*(2) = \max\begin{cases} r_2(0) + f_3^*(2) = 3.4 \\ r_2(1) + f_3^*(1) = 4 \\ r_2(2) + f_3^*(0) = 3.25 \end{cases} && x_2^*(2) = 1;\\
&f_2^*(3) = \max\begin{cases} r_2(0) + f_3^*(3) = 4.35 \\ r_2(1) + f_3^*(2) = 5.4 \\ r_2(2) + f_3^*(1) = 5.25 \\ r_2(3) + f_3^*(0) = 4.35 \end{cases} && x_2^*(3) = 1;\\
&f_2^*(4) = \max\begin{cases} r_2(1) + f_3^*(3) = 6.35 \\ r_2(2) + f_3^*(2) = 6.65 \\ r_2(3) + f_3^*(1) = 6.35 \end{cases} && x_2^*(4) = 2;\\
&f_2^*(5) = \max\begin{cases} r_2(2) + f_3^*(3) = 7.60 \\ r_2(3) + f_3^*(2) = 7.75 \end{cases} && x_2^*(5) = 3;\\
&f_2^*(6) = r_2(3) + f_3^*(3) = 8.7 && x_2^*(6) = 3.
\end{aligned}$$

Step 3: $i = 1$. The state $s_1 = 6$, and the decision x_1 is in the range $0, 1, 2, 3$. By a routine computation, we have

$$r_1(0) = 0, \quad r_1(1) = 2, \quad r_1(2) = 3.1, \quad r_1(3) = 4.2.$$

Thus, using the recursion (1), we have

$$f_1(6) = \max \begin{cases} r_1(0) + f_2^*(6) = 8.7 \\ r_1(1) + f_2^*(5) = 9.75 \\ r_1(2) + f_2^*(4) = 9.75 \\ r_1(3) + f_2^*(3) = 9.6 \end{cases} \qquad x_1^*(6) = 1 \text{ or } 2.$$

Therefore, the maximum expected revenue is 9.75 and hence the maximum expected profit is $9.53 - 6 = 3.53$ dollars. The optimal allocation is to assign store 1 $x_1^*(6) = 1$ gallon, assign store 2 $x_2^*(6-1) = 3$ gallons, and assign store 3 $x_3^*(6-1-3) = 2$ gallons, or assign store 1 $x_1^*(6) = 2$ gallons, assign store 2 $x_2^*(6-2) = 2$ gallons, and assign store 3 $x_3^*(6-2-2) = 2$ gallons.

1.2 Determining Reject Allowances

Example: The HIT-AND-MISS MANUFACTURING COMPANY has received an order to supply one item of a particular type. However, the customer has specified such stringent quality requirements that the manufacturer may have to produce more than one item to obtain an item that is acceptable. The number of extra items produced in a production run is called the reject allowance. Including a reject allowance is common practice when producing for a custom order, and it seems advisable in this case.

The manufacturer estimates that each item of this type that is produced will be acceptable with probability 1/2 and defective (not acceptable) with probability 1/2. Thus, the number of acceptable items produced in a lot of size L will have a binomial distribution; i.e., the probability of producing no acceptable items in such a lot is $(1/2)^L$.

Marginal production costs for this product are estimated to be \$100 per item (even if defective), and excess items are worthless. In addition, a setup cost of \$300 must be incurred whenever the production process is set up for this product, and a completely new setup at this same cost is required for each subsequent production run if a lengthy inspection procedure reveals that a completely lot has not yielded an acceptable item. The manufacturer has time to make no more than three production runs. If an acceptable item has not ben obtained by the end of the third production run, the cost to the manufacturer in lost sales income and penalty costs will be \$1,600.

The objective is to determine the policy regarding the lot size for the required production run(s) that minimizes total expected cost for the manufacturer. ⊠

Solution. The elements of dynamic programming are as follows:

1. Stage i is the production run i, $i = 1, 2, 3$;
2. Decision x_i is the lot size for stage i; i.e., the number of items to produce in stage i;
3. State s_i is the number of acceptable items still needed at beginning of stage i, $s_i = 0, 1$;
4. $f_i(s_i, x_i)$: minimum total expected cost for stages $i, i+1, \ldots 3$, given the state in stage i is s_i and a decision x_i is made in stage i;
5. $f_i^*(s_i)$: minimum total expected cost for stages $i, i+1, \ldots, 3$, given the state in stage i is s_i;
6. $x_i^*(s_i)$: the optimal decision in stage i, given that the state in stage i is s_i.

At stage 1, we have $s_1 = 1$. Also, notice that if $s_i = 0$, then $f_i^*(0) = 0$. This is because $s_i = 0$ means that an acceptable item has been produced before stage i, and hence no production cost will be incurred in stages $i, i+1, \ldots, 3$. The transition of the states between the stages is as follows. If $s_i = 0$, then we must have $s_{i+1} = 0$. On the other hand, if $s_i = 1$ and the decision at stage i is x_i, then we have

$$s_{i+1} = \begin{cases} 1, & \text{with probability } \left(\frac{1}{2}\right)^{x_i} \\ 0, & \text{with probability } 1 - \left(\frac{1}{2}\right)^{x_i}. \end{cases}$$

Let us denote by $C(x_i)$ the cost of a production with lot size x_i. By definition, we have

$$C(x_i) = \begin{cases} 0, & \text{if } x_i = 0 \\ 3 + x_i, & \text{if } x_i > 0. \end{cases} \tag{2}$$

Therefore, for $s_i = 1$,

$$\begin{aligned} f_i(1, x_i) &= \left(\frac{1}{2}\right)^{x_i} \cdot \left(C(x_i) + f_{i+1}^*(1)\right) + \left(1 - \left(\frac{1}{2}\right)^{x_i}\right) \cdot \left(C(x_i) + f_{i+1}^*(0)\right) \\ &= C(x_i) + \left(\frac{1}{2}\right)^{x_i} \cdot f_{i+1}^*(1) \end{aligned}$$

and the recursion of this DP problem is

$$f_i^*(0) = 0 \tag{3}$$

$$f_i^*(1) = \min_{x_i = 0,1,\ldots} \left\{ C(x_i) + \left(\frac{1}{2}\right)^{x_i} \cdot f_{i+1}^*(1) \right\}, \qquad i = 1, 2, 3, \tag{4}$$

where $f_4^*(1) = 16$ is the cost if no acceptable item is produced after 3 production runs.

Now we are ready to solve the problem using the working-backward procedure. Since $f_i^*(0) = 0$ for all $i = 1, 2, 3$, we only need to compute the values of $f_i^*(1)$ for $i = 1, 2, 3$.

Step 1: $i = 3$.

$$f_3^*(1) = \min_{x_3 = 0,1,\ldots} \left\{ C(x_3) + \left(\frac{1}{2}\right)^{x_3} \cdot f_4^*(1) \right\}.$$

Using $f_4^*(1) = 16$ and the formula for $C(\cdot)$ in (2), we have

$$C(0) + \left(\frac{1}{2}\right)^0 \cdot f_4^*(1) = 16$$

$$C(1) + \left(\frac{1}{2}\right)^1 \cdot f_4^*(1) = 12$$

$$C(2) + \left(\frac{1}{2}\right)^2 \cdot f_4^*(1) = 9$$

$$C(3) + \left(\frac{1}{2}\right)^3 \cdot f_4^*(1) = 8$$

$$C(4) + \left(\frac{1}{2}\right)^4 \cdot f_4^*(1) = 8$$

$$C(5) + \left(\frac{1}{2}\right)^5 \cdot f_4^*(1) = 8.5.$$

In addition, when $x_3 \geq 6$, we have

$$C(x_3) + \left(\frac{1}{2}\right)^{x_3} \cdot f_4^*(1) \geq C(x_3) \geq C(6) = 9.$$

Thus

$$f_3^*(1) = 8, \quad x_3^*(1) = 3 \text{ or } 4. \tag{5}$$

Step 2: $i = 2$.

$$f_2^*(1) = \min_{x_2=0,1,\ldots} \left\{ C(x_2) + \left(\frac{1}{2}\right)^{x_2} \cdot f_3^*(1) \right\}.$$

From (5), we have $f_3^*(1) = 8$. Thus

$$C(0) + \left(\frac{1}{2}\right)^{0} \cdot f_3^*(1) = 8$$
$$C(1) + \left(\frac{1}{2}\right)^{1} \cdot f_3^*(1) = 8$$
$$C(2) + \left(\frac{1}{2}\right)^{2} \cdot f_3^*(1) = 7$$
$$C(3) + \left(\frac{1}{2}\right)^{3} \cdot f_3^*(1) = 7$$
$$C(4) + \left(\frac{1}{2}\right)^{4} \cdot f_3^*(1) = 7.5$$

In addition, when $x_2 \geq 5$, we have

$$C(x_2) + \left(\frac{1}{2}\right)^{x_2} \cdot f_3^*(1) \geq C(x_2) \geq C(5) = 8.$$

Thus

$$f_2^*(1) = 7, \quad x_2^*(1) = 2 \text{ or } 3. \tag{6}$$

Step 3: $i = 1$.

$$f_1^*(1) = \min_{x_1=0,1,\ldots} \left\{ C(x_1) + \left(\frac{1}{2}\right)^{x_1} \cdot f_2^*(1) \right\}.$$

From (6), we have $f_2^*(1) = 7$. Thus

$$C(0) + \left(\frac{1}{2}\right)^{0} \cdot f_2^*(1) = 7$$
$$C(1) + \left(\frac{1}{2}\right)^{1} \cdot f_2^*(1) = 7.5$$
$$C(2) + \left(\frac{1}{2}\right)^{2} \cdot f_2^*(1) = 6.75$$
$$C(3) + \left(\frac{1}{2}\right)^{3} \cdot f_2^*(1) = 6.875$$
$$C(4) + \left(\frac{1}{2}\right)^{4} \cdot f_2^*(1) = 7.4375$$

In addition, when $x_1 \geq 5$, we have

$$C(x_1) + \left(\frac{1}{2}\right)^{x_1} \cdot f_2^*(1) \geq C(x_1) \geq C(5) = 8.$$

Thus

$$f_1^*(1) = 6.75, \quad x_1^*(1) = 2.$$

Therefore, the optimal policy is to produce two items on the first production run; if none is acceptable, then produce either two or three items on the second production run; if none is acceptable, then produce either three or four items on the third production run. The total expected cost for this policy is \$675.

1.3 Gambling Game

Example: A gambler has \$2. She is allowed to play a game of chance four times, and her goal is to maximize her probability of ending up with at least \$6. If the gambler bets b dollars on a play of the game, then with probability 0.4 she wins the game and increases her capital by b dollars; with probability 0.6 she loses the game and decreases her capital by b dollars. On any play of the game, the gambler may not bet more money than she has available. Determine a betting strategy that will maximize the gambler's probability of attaining a wealth of at least \$6 by the end of the fourth game. We assume that bets of zero dollars (that is, not betting) are allowed. ⊠

Solution. This problem aims at maximizing the probability of an event. For such problem, we can assign a reward of 1 if the favorable event (she has \$6 at then end of game 4) occurs and a reward of 0 if it does not occur. Then expected reward will be equivalent to the probability that the favorable event will occur. Thus the problem falls into the category of maximizing the expected reward problems.

The elements of dynamic programming are as follows:

1. Stage i is to start game i, $i = 1, 2, 3, 4$.
2. Decision x_i is the amount of money to bet in game i.
3. State s_i is the amount of money she has at the start of game i.
4. $f_i(s_i, x_i)$: maximum probability of attaining \$6 by the end of the fourth game, given the state in stage i is s_i and a decision x_i is made in stage i.
5. $f_i^*(s_i)$: maximum probability of attaining \$6 by the end of the fourth game, given the state in stage i is s_i.
6. $x_i^*(s_i)$: optimal decision in stage i, given the state in stage i is s_i.

The transition between states:

$$s_{i+1} = \begin{cases} s_i + x_i, & \text{with probability } 0.4 \\ s_i - x_i, & \text{with probability } 0.6 \end{cases}$$

and the decision x_i must satisfy $0 \leq x_i \leq s_i$. Moreover, we have

$$f_i(s_i, x_i) = 0.4 \cdot f_{i+1}^*(s_i + x_i) + 0.6 \cdot f_{i+1}^*(s_i - x_i)$$

and hence

$$f_i^*(s_i) = \max_{0 \leq x_i \leq s_i} \left\{ 0.4 \cdot f_{i+1}^*(s_i + x_i) + 0.6 \cdot f_{i+1}^*(s_i - x_i) \right\}.$$

Step 1: $i = 4$. The strategy for this stage is clear: if the gambler has \$6 or more, don't bet anything; if she has less than \$6, bet enough money to ensure that she will have \$6 if she wins the game. Note that if she begins game 4 with \$0, \$1, or \$2, there is no way to win. This reasoning lead to the following results:

$$\begin{array}{ll}
f_4^*(0) = 0 & x_4^*(0) = 0 \\
f_4^*(1) = 0 & x_4^*(1) = 0 \text{ or } 1 \\
f_4^*(2) = 0 & x_4^*(2) = 0, 1, \text{or } 2 \\
f_4^*(3) = 0.4 & x_4^*(3) = 3 \\
f_4^*(4) = 0.4 & x_4^*(4) = 2, 3, \text{or } 4 \\
f_4^*(5) = 0.4 & x_4^*(5) = 1, 2, 3, 4, \text{or } 5
\end{array}$$

For $s_4 \geq 6$,

$$f_4^*(s_4) = 1 \qquad x_4^*(s_4) = 0, 1, \ldots, (s_4 - 6).$$

Step 2: $i = 3$. For any s_3, we have the recursion

$$f_3^*(s_3) = \max_{0 \leq x_3 \leq s_3} \left\{ 0.4 \cdot f_4^*(s_3 + x_3) + 0.6 \cdot f_4^*(s_3 - x_3) \right\}.$$

Thus

$$f_3^*(0) = 0 \qquad x_3^*(0) = 0$$

$$f_3^*(1) = \max \begin{Bmatrix} 0.4f_4^*(1) + 0.6f_4^*(1) = 0 & \text{(Bet 0)} \\ 0.4f_4^*(2) + 0.6f_4^*(0) = 0 & \text{(Bet 1)} \end{Bmatrix} = 0 \qquad x_3^*(1) = 0 \text{ or } 1$$

$$f_3^*(2) = \max \begin{Bmatrix} 0.4f_4^*(2) + 0.6f_4^*(2) = 0 & \text{(Bet 0)} \\ 0.4f_4^*(3) + 0.6f_4^*(1) = 0.16 & \text{(Bet 1)} \\ 0.4f_4^*(4) + 0.6f_4^*(0) = 0.16 & \text{(Bet 2)} \end{Bmatrix} = 0.16 \qquad x_3^*(2) = 1 \text{ or } 2$$

$$f_3^*(3) = \max \begin{Bmatrix} 0.4f_4^*(3) + 0.6f_4^*(3) = 0.4 & \text{(Bet 0)} \\ 0.4f_4^*(4) + 0.6f_4^*(2) = 0.16 & \text{(Bet 1)} \\ 0.4f_4^*(5) + 0.6f_4^*(1) = 0.16 & \text{(Bet 2)} \\ 0.4f_4^*(6) + 0.6f_4^*(0) = 0.4 & \text{(Bet 3)} \end{Bmatrix} = 0.4 \qquad x_3^*(3) = 0 \text{ or } 3$$

$$f_3^*(4) = \max \begin{Bmatrix} 0.4f_4^*(4) + 0.6f_4^*(4) = 0.4 & \text{(Bet 0)} \\ 0.4f_4^*(5) + 0.6f_4^*(3) = 0.4 & \text{(Bet 1)} \\ 0.4f_4^*(6) + 0.6f_4^*(2) = 0.4 & \text{(Bet 2)} \\ 0.4f_4^*(7) + 0.6f_4^*(1) = 0.4 & \text{(Bet 3)} \\ 0.4f_4^*(8) + 0.6f_4^*(0) = 0.4 & \text{(Bet 4)} \end{Bmatrix} = 0.4 \qquad x_3^*(4) = 0, 1, 2, 3, \text{or } 4$$

$$f_3^*(5) = \max \begin{Bmatrix} 0.4f_4^*(5) + 0.6f_4^*(5) = 0.4 & \text{(Bet 0)} \\ 0.4f_4^*(6) + 0.6f_4^*(4) = 0.64 & \text{(Bet 1)} \\ 0.4f_4^*(7) + 0.6f_4^*(3) = 0.64 & \text{(Bet 2)} \\ 0.4f_4^*(8) + 0.6f_4^*(2) = 0.4 & \text{(Bet 3)} \\ 0.4f_4^*(9) + 0.6f_4^*(1) = 0.4 & \text{(Bet 4)} \\ 0.4f_4^*(10) + 0.6f_4^*(0) = 0.4 & \text{(Bet 5)} \end{Bmatrix} = 0.64 \qquad x_3^*(5) = 1 \text{ or } 2$$

For $s_3 \geq 6$,

$$f_3^*(s_3) = 1 \qquad x_3^*(s_3) = 0, 1, \ldots, (s_3 - 6).$$

Step 3: $i = 2$. For any s_2, we have the recursion

$$f_2^*(s_2) = \max_{0 \leq x_2 \leq s_2} \left\{ 0.4 \cdot f_3^*(s_2 + x_2) + 0.6 \cdot f_3^*(s_2 - x_2) \right\}.$$

Thus

$$f_2^*(0) = 0 \qquad x_2^*(0) = 0$$

$$f_2^*(1) = \max \begin{Bmatrix} 0.4f_3^*(1) + 0.6f_3^*(1) = 0 \quad \text{(Bet 0)} \\ 0.4f_3^*(2) + 0.6f_3^*(0) = 0.064 \quad \text{(Bet 1)} \end{Bmatrix} = 0.064 \qquad x_2^*(1) = 1$$

$$f_2^*(2) = \max \begin{Bmatrix} 0.4f_3^*(2) + 0.6f_3^*(2) = 0.16 \quad \text{(Bet 0)} \\ 0.4f_3^*(3) + 0.6f_3^*(1) = 0.16 \quad \text{(Bet 1)} \\ 0.4f_3^*(4) + 0.6f_3^*(0) = 0.16 \quad \text{(Bet 2)} \end{Bmatrix} = 0.16 \qquad x_2^*(2) = 0, 1, \text{or } 2$$

$$f_2^*(3) = \max \begin{Bmatrix} 0.4f_3^*(3) + 0.6f_3^*(3) = 0.4 \quad \text{(Bet 0)} \\ 0.4f_3^*(4) + 0.6f_3^*(2) = 0.256 \quad \text{(Bet 1)} \\ 0.4f_3^*(5) + 0.6f_3^*(1) = 0.256 \quad \text{(Bet 2)} \\ 0.4f_3^*(6) + 0.6f_3^*(0) = 0.4 \quad \text{(Bet 3)} \end{Bmatrix} = 0.4 \qquad x_2^*(3) = 0 \text{ or } 3$$

$$f_2^*(4) = \max \begin{Bmatrix} 0.4f_3^*(4) + 0.6f_3^*(4) = 0.4 \quad \text{(Bet 0)} \\ 0.4f_3^*(5) + 0.6f_3^*(3) = 0.496 \quad \text{(Bet 1)} \\ 0.4f_3^*(6) + 0.6f_3^*(2) = 0.496 \quad \text{(Bet 2)} \\ 0.4f_3^*(7) + 0.6f_3^*(1) = 0.4 \quad \text{(Bet 3)} \\ 0.4f_3^*(8) + 0.6f_3^*(0) = 0.4 \quad \text{(Bet 4)} \end{Bmatrix} = 0.496 \qquad x_2^*(4) = 1 \text{ or } 2$$

$$f_2^*(5) = \max \begin{Bmatrix} 0.4f_3^*(5) + 0.6f_3^*(5) = 0.64 \quad \text{(Bet 0)} \\ 0.4f_3^*(6) + 0.6f_3^*(4) = 0.64 \quad \text{(Bet 1)} \\ 0.4f_3^*(7) + 0.6f_3^*(3) = 0.64 \quad \text{(Bet 2)} \\ 0.4f_3^*(8) + 0.6f_3^*(2) = 0.496 \quad \text{(Bet 3)} \\ 0.4f_3^*(9) + 0.6f_3^*(1) = 0.4 \quad \text{(Bet 4)} \\ 0.4f_3^*(10) + 0.6f_3^*(0) = 0.4 \quad \text{(Bet 5)} \end{Bmatrix} = 0.64 \qquad x_2^*(5) = 0, 1, \text{or } 2$$

For $s_2 \geq 6$,

$$f_2^*(s_2) = 1 \qquad x_2^*(s_2) = 0, 1, \ldots, (s_2 - 6).$$

Step 4: $i = 1$. $s_1 = 2$. Thus

$$f_1^*(2) = \max \begin{Bmatrix} 0.4f_2^*(2) + 0.6f_2^*(2) = 0.16 \quad \text{(Bet 0)} \\ 0.4f_2^*(3) + 0.6f_2^*(1) = 0.1984 \quad \text{(Bet 1)} \\ 0.4f_2^*(4) + 0.6f_2^*(0) = 0.1984 \quad \text{(Bet 2)} \end{Bmatrix} = 0.1984 \qquad x_1^*(2) = 1 \text{ or } 2$$

Therefore the gambler has a 0.1984 change of reaching \$6. Suppose the gambler begins by letting $x_1^*(2) = 1$. Then Figure 2 (with each $b_i(\cdot)$ represents $x_i^*(\cdot)$) represents the optimal way to gamble in order to reach the highest probability of reaching \$6 at the end of the fourth game.

1.4 Parking Spaces

Example: Robert is trying to find a parking place near his favorite restaurant. He is approaching the restaurant from the west, and his goal is to park as nearby as possible. The available parking places are pictured in Figure 3. Robert is nearsighted and cannot see ahead; he can only see

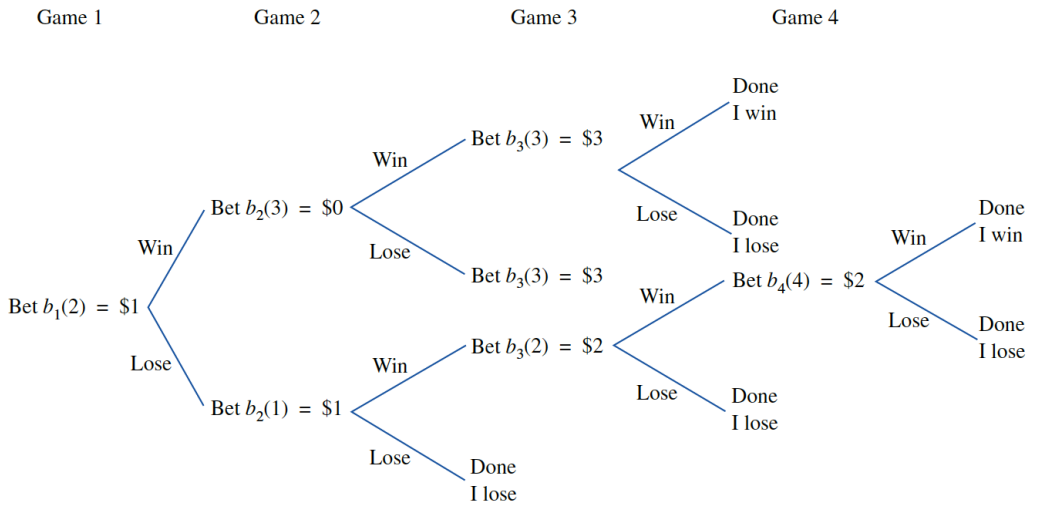

Figure 2: An optimal way to play the gamble game

$$\rightarrow \boxed{-T} \quad \boxed{1-T} \quad \boxed{2-T} \cdots \boxed{-2} \quad \boxed{-1} \quad \boxed{0} \quad \boxed{1} \quad \boxed{2} \cdots \boxed{T}$$

$$0 = \text{Restaurant}$$

Figure 3: Location of parking places

whether the space he is at now is empty. When Robert arrives at an empty space, he must decide whether to park there or to continue to look for a closer space. Once he passes a space, he cannot return to it. Robert estimates that the probability that space t is empty is p_t. If he does not end up with a parking space, he is embarrassed and incurs a cost M. If he does park in space t, he incurs a cost $|t|$ (the closer to the restaurant, the lower the cost is). Show how Robert can use dynamic programming to develop a parking strategy that minimizes his expected cost. ⊠

Solution. The elements of dynamic programming:

1. Stage i: Robert is at the parking space i, $i = -T, -T+1, \ldots, -1, 0, 1, \ldots, T-1, T$.
2. Decision x_i: to park or continue at stage i.
3. State s_i: the availability of parking space i, $s_i = 0$, or 1, with 0 representing not available and 1 representing available.
4. $f_i(s_i, x_i)$: the minimum expected cost if Robert is at space i, given that the availability at space i is s_i, and Robert makes decision x_i at space i.
5. $f_i^*(s_i)$: the minimum expected cost if Robert is at space i, given that the availability at space i is s_i.

6. $x_i^*(s_i)$: the optimal decision at space i, given that the availability at space i is s_i.

Clearly,

$$f_T^*(0) = M$$
$$f_T^*(1) = \min\{T, M\}$$

The relationship between $f_i(s_i, x_i)$ and $f_{i+1}^*(s_{i+1})$ is

$$f_i(1, \text{park}) = |i|$$
$$f_i(1, \text{cont.}) = p_{i+1} f_{i+1}^*(1) + (1 - p_{i+1}) f_{i+1}^*(0)$$
$$f_i(0, \text{cont.}) = p_{i+1} f_{i+1}^*(1) + (1 - p_{i+1}) f_{i+1}^*(0)$$

Thus, the recursion between i and $i+1$ stages becomes

$$f_i^*(0) = f_i(0, \text{cont.}) = p_{i+1} f_{i+1}^*(1) + (1 - p_{i+1}) f_{i+1}^*(0)$$
$$f_i^*(1) = \min\{f_i(1, x_i) : x_i = \text{park or cont.}\} = \min \begin{cases} |i| & \text{(Park)} \\ p_{i+1} f_{i+1}^*(1) + (1 - p_{i+1}) f_{i+1}^*(0) & \text{(Cont.)} \end{cases}$$

In what follows, we suppose $T = 2$ and the details are given in Table 4. And we consider the

Parking Space	-2	-1	0	1	2
Prob. of empty	0.5	0.2	0.1	0.2	0.5
Parking Cost	2	1	0	1	2

Table 4: Parking spaces, probability of availability, and parking costs

two cases (i) $M = 3$ and (ii) $M = 10$. Intuitively, a larger M will give Robert the intention to park earlier. We will see how this is reflected by DP procedures.

Case (i): $M = 3$

At parking space 2, we have

$$f_2^*(0) = 3$$
$$f_2^*(1) = \min\{2, 3\} = 2$$
$$x_2^*(1) = \text{park}$$

At parking space 1, we have

$$f_1^*(0) = 0.5 \times f_2^*(0) + 0.5 \times f_2^*(1) = 2.5$$
$$f_1^*(1) = \min\{1, 2.5\} = 1$$
$$x_1^*(1) = \text{park}$$

At parking space 0, we have

$$f_0^*(0) = 0.8 \times f_1^*(0) + 0.2 \times f_1^*(1) = 2 + 0.2 = 2.2$$
$$f_0^*(1) = \min\{0, 2.2\} = 0$$
$$x_0^*(1) = \text{park}$$

At parking space -1, we have

$$\begin{aligned} f^*_{-1}(0) &= 0.9 \times f^*_0(0) + 0.1 \times f^*_0(1) = 1.98 \\ f^*_{-1}(1) &= \min\{1, 1.98\} \\ x^*_{-1}(1) &= \text{park} \end{aligned}$$

At parking space -2, we have

$$\begin{aligned} f^*_{-2}(0) &= 0.8 \times f^*_{-1}(0) + 0.2 \times f^*_{-1}(1) = 1.784 \\ f^*_{-2}(1) &= \min\{2, 1.784\} = 1.784 \\ x^*_{-2}(1) &= \text{cont.} \end{aligned}$$

Therefore, in this case, the minimum expected cost is 1.784. The optimal parking strategy is: do not park at parking space -2, even it is available; after passing space -2, park at the first available parking space.

Case (ii): $M = 10$.

At parking space 2, we have

$$\begin{aligned} f^*_2(0) &= 10 \\ f^*_2(1) &= \min\{2, 10\} = 2 \\ x^*_2(1) &= \text{park} \end{aligned}$$

At parking space 1, we have

$$\begin{aligned} f^*_1(0) &= 0.5 \times f^*_2(0) + 0.5 \times f^*_2(1) = 6 \\ f^*_1(1) &= \min\{1, 6\} = 1 \\ x^*_1(1) &= \text{park} \end{aligned}$$

At parking space 0, we have

$$\begin{aligned} f^*_0(0) &= 0.8 \times f^*_1(0) + 0.2 \times f^*_1(1) = 5 \\ f^*_0(1) &= \min\{0, 5\} = 0 \\ x^*_0(1) &= \text{park} \end{aligned}$$

At parking space -1, we have

$$\begin{aligned} f^*_{-1}(0) &= 0.9 \times f^*_0(0) + 0.1 \times f^*_0(1) = 4.5 \\ f^*_{-1}(1) &= \min\{1, 4.5\} = 1 \\ x^*_{-1}(1) &= \text{park} \end{aligned}$$

At parking space -2, we have

$$\begin{aligned} f^*_{-2}(0) &= 0.8 \times f^*_{-1}(0) + 0.2 \times f^*_{-1}(1) = 3.8 \\ f^*_{-2}(1) &= \min\{2, 3.8\} = 2 \\ x^*_{-2}(1) &= \text{park} \end{aligned}$$

Therefore, the optimal decision in this case is to park at the first available parking space. Since parking space -2 has 0.5 probability of being empty, the expected parking cost in this case is

$$0.5 \times f^*_{-2}(0) + 0.5 \times f^*_{-2}(1) = 2.9$$

2 References

- Hillier and Lieberman, *Introduction to Operations Research*, Chapter 11.4;
- Winston, *Operations Research: Applications and Algorithms*, Chapter 19.