Solved Examples for Chapter 11

Example 1 for Section 11.3

The Build-Em-Fast Company has agreed to supply its best customer with three widgits during *each* of the next 3 weeks, even though producing them will require some overtime work. The relevant production data are as follows:

Week	Maximum Production, Regular Time	Maximum Production, Overtime	Production Cost per Unit Regular Time
1	2	2	\$300
2	3	2	\$500
3	1	2	\$400

The cost per unit produced with overtime for each week is \$100 more than for regular time. The cost of storage is \$50 per unit for each week it is stored. There is already an inventory of two widgets on hand currently, but the company does not want to retain any widgets in inventory after the 3 weeks.

Management wants to know how many units should be produced in each week to minimize the total cost of meeting the delivery schedule.

Let us solve this by using dynamic programming.

Application of Dynamic Programming

The decisions that need to be made are the number of units to produce each week, so the decision variables are

x_n = number of widgets to produce in week n, for n = 1, 2, 3.

To choose the value of x_n, it is necessary to know the number of widgets already on hand at the beginning of week n. Therefore, the "state of the system" then is

s_n = number of widgets on hand at the beginning of week n, for n = 1, 2, 3.

Because of the shipment of three widgets to the customer each week, note that

$$s_{n+1} = s_n + x_n - 3.$$

Also note that two widgets are on hand at the beginning of week 1, so

$$s_1 = 2.$$

To introduce symbols for the data given in the above table for each week n, let

c_n = unit production cost in regular time,

r_n = maximum regular time production,
m_n = maximum total production.

The corresponding data are given in the following table.

n	r_n	m_n	c_n
1	2	4	300
2	3	5	500
3	1	3	400

We wish to minimize the total cost of meeting the delivery schedule, so our measure of performance is total cost. For n = 1, 2, 3, let

$p_n(s_n, x_n)$ = cost incurred in week n when the state is s_n and the decision is x_n.

Thus,

$$p_n(s_n, x_n) = c_n x_n + 100 \max(0, x_n - r_n) + 50 \max(0, s_n + x_n - 3) .$$

Using the notation in Sec. 11.2, let

$f_n^*(s_n)$ = optimal cost for week n onward (through the end of week 3) when starting week n in state s_n, for n = 1, 2, 3.

Given s_n, the feasible values of x_n are the integers such that $x_n \le m_n$ and $x_n \ge 3 - s_n$ (assuming $s_n \le 3$) in order to provide the customer with 3 widgets in week n. Thus, because $s_{n+1} = s_n + x_n - 3$, the optimal value of x_n (denoted by x_n^*) is obtained from the following recursive relationship.

$$f_n^*(s_n) = \min_{3-s_n \le x_n \le m_n} [p_n(s_n, x_n) + f_{n+1}^*(s_n + x_n - 3)],$$

where

$f_4^*(s_4) = 0$ for $s_4 = 0$.

Recall that the company does not want to retain any widgets in inventory after the three weeks, so $s_4 = 0$. Therefore, the optimal policy for week 3 obviously is to produce just enough widgets to have a total of three to ship to the customer, so

$x_3^* = 3 - s_3$.

Given x_3^* and $f_3^*(s_3)$, we can use the recursive relationship to solve for the optimal policy for week 2 and then for week 1. These calculations are shown below.

For *n* = 3:

s_3	$f_3^*(s_3)$	x_3^*
0	1,400	3
1	900	2
2	400	1
$3 \le s_3$	0	0

For *n* = 2:

s_2 \ x_2	$f_2(s_2, x_2) = p_2(s_2, x_2) + f_3^*(s_2 + x_2 - 3)$						$f_2^*(s_2)$	x_2^*
	0	1	2	3	4	5		
0				$\boxed{2,900}$	3,050	3,200	2,900	3
1			$\boxed{2,400}$	2,450	2,600	2,850	2,400	2
2		$\boxed{1,900}$	1,950	2,000	2,250	2,900	1,900	1
3	$\boxed{1,400}$	1,450	1,500	1,650	2,300	2,950	1,400	0

For *n* = 1:

s_1 \ x_1	$f_1(s_1, x_1) = p_1(s_1, x_1) + f_2^*(s_1 + x_1 - 3)$					$f_1^*(s_1)$	x_1^*
	0	1	2	3	4		
2		3,200	3,050	3,000	$\boxed{2,950}$	2,950	4

Hence, the optimal plan is to produce four widgets in period 1, store three of them until period 2, and produce three in period 3, with a total cost of $2,950.

Example 2 for Section 11.3

Consider the following nonlinear programming problem:

Maximize $\quad Z = x_1^2 x_2$,
subject to

$$x_1^2 + x_2^3 \le 2.$$

(There are no nonnegativity constraints.) Let us use dynamic programming to solve this problem.

Application of Dynamic Programming

Let x_1 be the decision variable at stage 1 and x_2 be the decision variable at stage 2.

The key to applying dynamic programming to this problem is to interpret the right-hand side of the constraint as the amount of a resource being made available to activities 1 and 2 (whose levels are x_1 and x_2). Then the state of the system entering stage 1 (before choosing x_1 and x_2) and entering stage 2 (before choosing x_2) is the amount of the resource still available for allocation to the remaining activities. Consequently, letting s_n denote the state of the system entering stage n, we have

$$s_1 = 2 \quad \text{and} \quad s_2 = 2 - x_1^2.$$

For n = 2:

At stage 2, we solve the following problem with variable x_2:

Maximize $f_2(s_2, x_2) = x_2$,

subject to

$$x_2 \le s_2.$$

The optimal solution is $x_2^* = s_2$, with $f_2^*(s_2) = s_2$.

For n = 1:

At stage 1, we maximize $f_1(2, x_1) = x_1^2(f_2^*(2-x_1^2)) = x_1^2(2-x_1^2)$,
subject to

$$x_1^2 \le 2.$$

$$\frac{\partial f_1(2,x_1)}{\partial x_1} = -4x_1^3 + 4x_1 = 0 \;\Rightarrow\; x_1 = 1,\ -1 \quad \text{are local maxima.}$$

$$\frac{\partial^2 f_1(2,x_1)}{\partial x_1^2} = 4(1-3x_1^2) < 0 \text{ for } x_1 = 1, -1.$$

Hence, $f_1(2,x_1)$ is locally concave around $x_1 = 1$ and $x_1 = -1$.

Since $f_1(2,1) = f_1(2,-1) = 1$, whereas $f(2, \sqrt{2}) = 0$ at the endpoints of the feasible region, we have both $x_1^* = 1$ and $x_1^* = -1$ as global maximizers and $f_1^*(2) = 1$.
Hence, the optimal solutions are $(x_1^*, x_2^*) = (1, 1)$, and $(x_1^*, x_2^*) = (1, -1)$ with an optimal objective function value of Z = 1.