Operations Analysis II

IE 322

Dynamic Programming

Dynamic Programming (DP) determines the optimum solution of a problem by decomposing it into stages.

1-Each Stage includes a simple variable sub problem.
2-Recursive equation linking the different stages.
3-Guarantees that each stage's optimal solution is also optimal for the entire problem.

Recursive Nature of Computations in DP is done in the sense that the optimum solution of one sub problem is used as an input to the next sub problem. By the time the last sub problem is solved, the optimum solution for the entire problem is at hand. The manner in which the recursive computations are carried out depends on how we decompose.

Forward Strategy

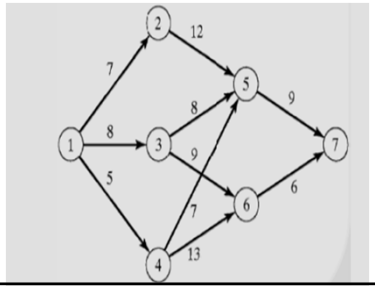
1- Divide the original problem into sub problems called stages.
2-Solve the first stage of the problem for all possible conditions or states.
3-Working forward from that first stage, solve each intermediate stage.
4-Obtain the optimal solution for the original problem by solving all stages sequentially.

Backward Strategy

1- Divide the original problem into sub problems called stages.
2-Solve the last stage of the problem for all possible conditions or states.
3-Working backward from that last stage, solve each intermediate stage.
4-Obtain the optimal solution for the original problem by solving all stages sequentially.

Shortest Path Problem

An industrial engineer is planning the production of a single product. The product can be manufactured in several ways and by several departments. The industrial engineer needs to find the least time to manufacture the product. The sequences of the operations are shown in the figure.
The arcs represent the operations.
The product has to pass through 3 operations.



This problem can be solved as finding the shortest route from
node 1 to node 7.
The problem can be decomposed into stages, where each stage represents an operation.
Decomposition is performed so that the shortest time to all the terminal nodes of a stage are used in the following stage.
The terminal nodes of the first stage are 2, 3, and 4.
The shortest time of the terminal does to node 1 are:
Shortest time from 1 to 2 = 7
Shortest time from 1 to 3 = 8
Shortest 1 to 4 = 5

The terminal nodes of the second stage are 5 and 6.
Using the shortest times from the previous stage, the shortest times to the terminal nodes are:
Shortest time from 1 to 5 = min{15+7, 8+8, 7+5} = 12
Shortest time from 1 to 6 = min{8+9, 5+13} = 17

The third stage has only one terminal node, node 7.
Shortest time from 1 to 7 = min{12+9, 17+6} = 21

Since node 7 is the last node, the solution has been determined, which is, the shortest time to manufacture the product is 21 minutes and the operations are (1,4), (4,5), and(5,7).

Forward Equations

To define the recursive equation for this manufacturing sequence problem, define $f_t(i)$ be the shortest time from node 1 to node i at stage t.
Define c_{ji} as the time from node j to node i.
Then, f_t is computed from f_{t-1} using the recursive formula:
$f_t(i) = min \{cji + f_{t-1}(j)\}$
$f_0(1) = 0$
In dynamic programming, 'i' is referred to as the state of the system at stage t.
In general, a state is the information that is needed at any stage to make an optimal decision.

Backward Equations

To define the recursive equation for this manufacturing sequence problem, define $f_t(i)$ be the shortest time from node 7 to node i at stage t.
Define c_{ij} as the time from node j to node i.
Then, f_t is computed from f_{t-1} using the recursive formula:
$f_t(i) = min \{cij + f_{t+1}(i)\}$
$f_4(7) = 0$
In dynamic programming, 'i' is referred to as the state of the system at stage t.
In general, a state is the information that is needed at any stage to make an optimal decision.

Solve Example 10.1-1, assuming that the following routes are used:
d(1,2) = 5, d(1,3) = 9, d(1,4) = 8
d(2,5) = 10, d(2,6) = 17
d(3,5) = 4, d(3,6) = 10
d(4,5) = 9, d(4,6) = 9
d(5,7) = 8
d(6,7) = 9

How to define stages and states in the Shortest Path Problem

In the shortest path problem we decompose the graph into stages. Each stage includes two sets of nodes (starting nodes and arrival nodes) such that the arrival nodes can be reached only from the starting nodes.

The arrival nodes of the current stage represent the different states of the current stage

The starting nodes of the current stage represent the states of the previous stage.

The set of starting nodes in the first stage includes only the source node (s)

The set of arrival nodes in the last stage includes only the sink node (t)

Knap Sack Problem

This classical problem deals with the situation in which a hiker must decide on the most valuable items to carry in a backpack.
There are n items 1,2….n.
There are m_i *unit of items i. The weight per unit of* item i is w_i *and* r_i *is the revenue per unit of item i.*
The hiker can carry a weight of at most *W.*

Maximize $z = r_1m_1 + r_2m_2 + \dots + r_nm_n$
Subject to
$w_1m_1 + w_2m_2 + \dots + w_nm_n <= W$
$m_1, m_2, \dots, m_n >= 0$ and Integer.

The three elements of the model are

1- stages, each stage is represented by item i, , i=1,2…n.
2-The alternatives at stage i are represented by the number mi of item i included in the knapsack. The associated return is r_im_i.
(Note that m_i can take values 0,1,….[W/wi])
3-The State at stage I is represented by x_i, the total weight assigned to stages i, i+1, … n.

Define : $f_i(x_i)$ = maximum return of stages i, i+1, … n at state x_i.

The recursive equation is

$$f_i(x_i) = \max_{\substack{m_i = 0,1,\dots\left[\frac{W}{w_i}\right] \\ x_i \le W}} \{r_i m_i + f_{i+1}(x_i - m_i w_i)\},\ i = 1,2,\dots. n$$

$$f_{n+1}(x_{n+1}) = 0$$

- A workstation in a production line has a time cycle of 4 minutes (production will produce the same products every cycle).
- The products, production times, and unit revenues are

Product i	w_i	r_i
1	2	31
2	3	47
3	1	14

- Which and how many of the products should be produced?

- At the beginning, the allocation of the time to the three products is not known.
- The best time allocations are calculated for all possible states, and this information is passed to the following stage.
- The solution will start from stage 3 and work backward.
- The recursive equation for stage 3 is

$$f_3(y_3) = \max_{\substack{m_3=0,1,\ldots,4/1 \\ y_3 \le 4}} \{14m_3\}$$

- The states in stage 3 are 5, because the number of items of product i can be 0, 1, …,or 4.

The computations on the recursive equation are

	$14m_i$						
y_3	$m_3 = 0$	$m_3 = 1$	$m_3 = 2$	$m_3 = 3$	$m_3 = 4$	$f_3(y_3)$	m_3
0	0	-	-	-	-	0	0
1	0	14	-	-	-	14	1
2	0	14	28	-	-	28	2
3	0	14	28	42	-	42	3
4	0	14	28	42	56	56	4

- For product 2, the maximum number that can be produced is 1.
- Hence, the alternatives at stage 2 are 0 and 1.
- The recursive equation is

$$f_2(y_2) = \max_{\substack{m_2=0,1 \\ y_2 \le 4}} \{47m_2 + f_3(y_2 - 3m_2)\}$$

- The computation are organized as follows

	$47m_2 + f_3(x_2 - 3m_2)$			
y_2	$m_2 = 0$	$m_2 = 1$	$f_2(y_2)$	m_2
0	$47(0) + f_3(0 - 3(0)) = 0$	-	0	0
1	$47(0) + f_3(1 - 3(0)) = 14$	-	14	0
2	28	-	28	0
3	42	47	47	1
4	56	61	61	1

- For product 1, the maximum number that can be produced is 2.
- Hence, the alternatives at stage 2 are 0, 1, and 2.
- The recursive equation is

$$f_1(y_1) = \max_{\substack{m_1=0,1,2 \\ y_1 \le 4}} \{31m_1 + f_2(y_1 - 2m_1)\}$$

	$31m_1 + f_1(x_1 - 2m_1)$				
y_2	$m_1 = 0$	$m_1 = 1$	$m_1 = 2$	$f_1(y_1)$	m_1
0	0	-	-	0	0
1	14	-	-	14	0
2	28	31	-	31	1
3	47	45	-	47	0
4	61	59	62	62	2

- The optimal solution is found by working backward.
- From stage 1, the optimum alternative is $m_1 = 2$.
- The allocated time for product 2 is $y_2 = y_1 - 2m_1 = 4 - 2 \times 2 = 0$.
- From stage 2, for $y_2 = 0$, the optimal $m_2 = 0$.
- The allocated time for product 3 is $y_3 = y_2 - 3m_2 = 0$.
- From stage 3, for $y_3 = 0$, the optimal $m_3 = 0$.
- The optimal production mix is to produce 2 units of product 1 only.

- **Problem 2(a) Problem set 10.3A page 412**
- Solve the knapsack problem when

$$w_1 = 4, r_1 = 70, w_2 = 1, r_2 = 20, w_3 = 2, r_3 = 40, W = 6.$$

Stage 3

m_3 can assume values 0,1,2,3

An alternative is feasible only if

$$w_3 m_3 \leq x_3$$

Thus we get the following table which gives the optimal return for each value of x_3:

x_3 \ m_3	0	1	2	3	$f_3(x_3)$	m_3^*
0	0	-	-	-	0	0
1	0	-	-	-	0	0
2	0	40	-	-	40	1
3	0	40	-	-	40	1
4	0	40	80	-	80	2
5	0	40	80	-	80	2
6	0	40	80	120	120	3

Stage 2. $f_2(x_2)=\max_{m_2}\{20m_2+f_3(x_2-m_2)\}$ max $m_2=[6/1]=6$

$20m_2+f_3(x_2-m_2)$

x_2	m_2=0	1	2	3	4	5	6	$f_2(x_2)$	m_2^*
0	0	-	-	-	-	-	-	0	0
1	0	20	-	-	-	-	-	20	1
2	0+40 = 40	20+0 = 20	40+0 = 40	-	-	-	-	40	0 or 2
3	0+40 = 40	20+40 = 60	40+0 = 40	60+0 = 60	-	-	-	60	1 or 3
4	0+80 = 80	20+40 = 60	40+40 = 80	60+0 = 60	80+0 = 80	-	-	80	0 or 2 or 4
5	0+80 = 80	20+80 = 100	40+40 = 80	60+40 = 100	80+0 = 80	100+0 = 100	-	100	1 or 3 or 5
6	0+120 = 120	20+80 = 100	40+80 = 120	60+40 = 100	80+40 = 120	100+0 = 100	120+0 = 120	120	0 or 2 or 4 or 6

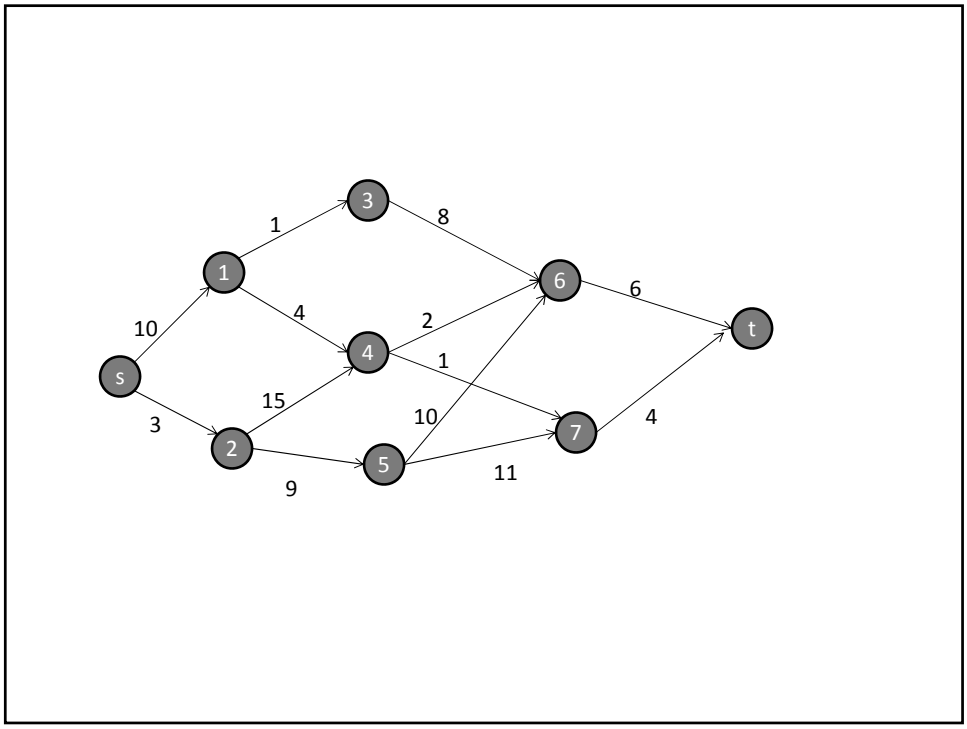
Stage 1 $f_1(x_1)=\max_{m_1}\{70m_1+f_2(x_1-4m_1)\}$ max $m_1=[6/4]=1$

$70m_1 + f_2(x_1 - 4m_1)$

x_1	m_1=0	m_1=1	$f_1(x_1)$	m_1^*
0	0+0 = 0	-	0	0
1	0+20 = 20	-	20	0
2	0+40 = 40	-	40	0
3	0+60 = 60	-	60	0
4	0+80 = 80	70+0 = 70	80	0
5	0+100 = 100	70+20 = 90	100	0
6	0+120 = 120	70+40 = 110	120	0

Optimal allocation:

$$m_1 = 0, m_2 = 0, m_3 = 3 \quad or$$
$$m_1 = 0, m_2 = 2, m_3 = 2 \quad or$$
$$m_1 = 0, m_2 = 4, m_3 = 1 \quad or$$
$$m_1 = 0, m_2 = 6, m_3 = 0$$


3
1
8
1
6
6
t
10
4
2
4
1
s
15
10
4
3
2
7
5
11
9