Dynamic Programming

11.1. INTRODUCTION

In previous chapters, we have seen how to solve the problems, where decision is made in single stage, *i.e.* one time period. But we may come across situations, where we may have to make decision in multistage, *i.e.* optimization of multistage decision problems. Dynamic programming is a technique for getting solutions for multistage decision problems. A problem, in which the decision has to be made at **successive stages,** is called a **multistage decision problem**. In this case, the problem solver will take decision at every stage, so that the total effectiveness defined over all the stages is optimal. Here the original problem is broken down or decomposed into small problems, which are known as **sub problems** or **stages** which is much convenient to handle and to find the optimal stage. For example, consider the problem of a sales manager, who wants to start from his head office and tour various branches of the company and reach the last branch. He has to plan his tour in such a way that he has to visit more number of branches and cover less distance as far as possible. He has to divide the network of the route connecting all the branches into various stages and workout, which is the best route, which will help him to cover more branches and less distance. We can give plenty of business examples, which are multistage decision problems. The technique of Dynamic programming was developed by Richard Bellman in the early 1950.

The computational technique used is known as **Dynamic Programming** or **Recursive Optimization**. We do not have a standard mathematical formulation of the Dynamic Programming Problem (D.P.P). For each problem, depending on the variables given, and objective of the problem, one has to develop a particular equation to fit for situation. Though we have quite good number of dynamic programming problems, sometimes to take advantage of dynamic programming, we introduce multistage nature in the problem and solve it by dynamic programming technique. Nowadays, application of Dynamic Programming is done in almost all day to day managerial problems, such as, inventory problems, waiting line problems, resource allocation problems etc. Dynamic programming problem may be classified depending on the following conditions.

(*i*) Dynamic programming problems may be classified depending on the nature of data available as *Deterministic and Stochastic or Probabilistic models*. In deterministic models, the outcome at any decision stage is unique, determined and known. In Probabilistic models, there is a set of possible outcomes with some probability distribution.

(*ii*) The possible decisions at any stage, from which we are to choose one, are called '**states**'. These may be finite or infinite. States are the possible situations in which the system may be at any stage.

(*iii*) Total number of stages in the process may be finite or infinite and may be known or unknown.

Now let us try to understand certain terms, which we come across very often in this chapter.

Stage: A stage signifies a portion of the total problem for which a decision can be taken. At each stage there are a number of alternatives, and the best out of those is called **stage decision**, which may be optimal for that stage, but contributes to obtain the optimal decision policy.

State: The condition of the decision process at a stage is called its state. The variables, which specify the condition of the decision process, *i.e.* describes the **status** of the system at a particular stage are called **state variables**. The number of state variables should be as small as possible, since larger the number of the state variables, more complicated is the decision process.

Policy: A rule, which determines the decision at each stage, is known as Policy. A policy is optimal one, if the decision is made at each stage in a way that the result of the decision is optimal over all the stages and not only for the current stage.

Principle of Optimality: Bellman's Principle of optimality states that "**An optimal policy (a sequence of decisions) has the property that whatever the initial state and decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.**"

This principle implies that a wrong decision taken at a stage does not prevent from taking optimal decision for the reaming stages. This principle is the firm base for dynamic programming technique. In the light of this, we can write a recurrence relation, which enables us to take the optimal decision at each stage.

Steps in getting the solution for dynamic programming problem:

- Mathematical formulation of the problem and to write the recursive equation (recursive relation connecting the optimal decision function for the 'n' stage problem with the optimal decision function for the $(n-1)$ stage subproblems).
- To write the relation giving the optimal decision function for one stage subproblem and solve it.
- To solve the optimal decision function for 2-stage, 3-stage …….. $(n-1)$ stage and then n-stage problem.

11.2. COMPUTATIONAL PROCEDURE IN DYNAMIC PROGRAMMING

Discrete or Continuous systems: There are two ways of solving (computational procedure) recursive equations depending on the type of the system. If the system is continuous one the procedure is different and if the system is discrete, we use a different method of computation. If the system is discrete, a tabular computational scheme is followed at each stage. The number of rows in each table is equal to the number of corresponding feasible state values and the number of columns is equal to the number of possible decisions. In case of continuous system, the optimal decision at each stage is obtained by using the usual classical technique such as differentiation etc.

- **Forward and Backward Equations:** If there are 'n' stages, and recursive equations for each stage is $f_1, f_2 \ldots\ldots f_n$ and if they are solved in the order f_1 to f_n and optimal return for f_1 is r_1 and that of f_2 is r_2 and so on, then the method of calculation is known as **forward computational procedure**.

- On the other hand, if they are solved in the order from $f_n, f_{n-1}, \ldots . f_1$, then the method is termed as **backward computational procedure**. (e.g. Solution to L.P.P. by dynamic programming).

The Algorithm

- Identify the decision variables and specify objective function to be optimized under certain limitations, if any.
- Decompose or divide the given problem into a number of smaller sub-problems or stages. Identify the state variables at each stage and write down the transformation function as a function of the state variable and decision variables at the next stage.
- Write down the general recursive relationship for computing the optimal policy. Decide whether forward or backward method is to follow to solve the problem.
- Construct appropriate stage to show the required values of the return function at each stage.
- Determine the overall optimal policy or decisions and its value at each stage. There may be more than one such optimal policy.

11.3. CHARACTERISTICS OF DYNAMIC PROGRAMMING

The basic features, which characterize the dynamic programming problem, are as follows:

(*i*) Problem can be sub-divided into stages with a policy decision required at each stage. A stage is a device to sequence the decisions. That is, it decomposes a problem into sub-problems such that an optimal solution to the problem can be obtained from the optimal solution to the sub-problem.

(*ii*) Every stage consists of a number of states associated with it. The states are the different possible conditions in which the system may find itself at that stage of the problem.

(*iii*) Decision at each stage converts the current stage into state associated with the next stage.

(*iv*) The state of the system at a stage is described by a set of variables, called **state variables**.

(*v*) When the current state is known, an optimal policy for the remaining stages is independent of the policy of the previous ones.

(*vi*) To identify the optimum policy for each state of the system, a recursive equation is formulated with 'n' stages remaining, given the optimal policy for each stage with $(n - 1)$ stages left.

(*vii*) Using recursive equation approach each time the solution procedure moves backward, stage by stage for obtaining the optimum policy of each stage for that particular stage, still it attains the optimum policy beginning at the initial stage.

11.4. PROBLEMS

Problem 11.1. (Product allocation problem)

A company has 8 salesmen, who have to be allocated to four marketing zones. The return of profit from each zone depends upon the number of salesmen working that zone. The expected returns for different number of salesmen in different zones, as estimated from the past records, are given below. Determine the optimal allocation policy.

	SALES	MARKETING IN	ZONES Rs. X 000	
No. of . Salesmen	Zone 1	Zone 2	Zone 3	Zone 4
0	45	30	35	42
1	58	45	45	54
2	70	60	52	60
3	82	70	64	70
4	93	79	72	82
5	101	90	82	95
6	108	98	93	102
7	113	105	98	110
8	118	110	100	110

Solution

The problem here is how many salesmen are to be allocated to each zone to maximize the total return. In this problem each zone can be considered as a **stage**, number of salesmen in each stage as **decision** variables. Number of salesmen available for allocation at a stage is the **state variable** of the problem.

Here let us consider the first stage (zone 1) and add to it the second stage (zone 2) and see what will be the optimal return and optimal allocation. Remember, that allocation of salesmen for each zone may be 0, 1, 2, …and 8. See the table below to understand how we can allocate salesmen between zones 1 and 2.

In this problem, decision policy requires making four interrelated decisions. What should be the number of salesmen in each of the four marketing zones? If x_1, x_2, x_3 and x_4 are the number of salesmen allocated to the four zones and $f_1(x_1)$,……$f_4(x_4)$ are respectively the returns from the four zones, then the objective function is

Maximize $Z = f_1(x_1) + f_2(x_2) + f_3(x_3) + f_4(x_4)$

Subject to: $x_1 + x_2 + x_3 + x_4 \leq 8$ and x_1, x_2, x_3 and x_4 are non-negative integers.

Or can be written as: Maximize $Z = \sum_{i=1}^{4} f_i(x_i)$ s.t. $\sum_{i=1}^{4} x_i = 8$ where all x_i are nonnegative integers.

No. of Salesmen in zone 1.	0	1	2	3	4	5	6	7	8
No. of Salesmen in zone 2.	8	7	6	5	4	3	2	1	0

Construct a table to calculate the return from the above combination.

Zone 1 → Salesmen		0	1	2	3	4	5	6	7	8
Return		45	58	70	82	93	101	108	113	118
Zone 2 ↓ Salesmen	Return									
0	30	**75**	88	100	112	123	141	138	143	148
1	45	**90**	103	115	127	138	146	153	158	
2	60	**105**	**118**	**130**	**142**	**153**	161	168		
3	70	115	128	140	152	**163**	171			
4	79	124	137	149	161	**172**				
5	90	135	148	160	**172**					
6	98	143	156	168						
7	105	150	163							
8	110	155								

Procedure: If we want to allocate zero salesmen, then zero to zone 1 and zero to zone 2 and the total outcome is 30 + 45 = Rs. 75 × 1000. This is written in the table where lines from zero from zone 1 and zone 2 intersect. As this is the only entry in the diagonal line it is made bold.

When company wants to allocate 1 salesman to two zones, the allocation is zero to zone 1 and 1 to zone 2 or 1 to zone 1 and zero to zone 2. The outcomes are entered where the horizontals from zone 2 and verticals from zone 1 intersect. Higher number is written in bold numbers. In this example, the outcomes are 90 and 88, 90 is written in bold. Similarly we have to allocate 8 salesmen and write the outcomes and bold the highest outcome in the diagonal. Sometimes, it may happen that there may be two or more same numbers indicating highest outcome. All these are written in bold letter. (Note: Instead on writing highest in bold letter, we can encircle the element or enclose it in a square or superscribe with a star.)

Now let us write the outcomes below:

Number of salesmen.	0	1	2	3	4	5	6	7	8	
Zone 1	0	0	0	1	2	3	4	4	4	3
Zone 2	0	1	2	2	2	2	2`	3	4	5
Outcome in Rs. × 1000	75	90	105	118	130	142	153	162	172	172

Now in the second stage, let us combine zone 3 and zone 4 and get the total market returns.

Combination of zone 3 and zone 4:

Zone 3 Salesmen →		0	1	2	3	4	5	6	7	8
Return.		35	45	52	64	72	82	93	98	100
Zone 4 Salesmen	Return. ↓									
0	42	**77**	**97**	94	**106**	114	124	136	140	142
1	54	89	**99**	**106**	**118**	126	136	**147**	152	
2	60	95	105	112	124	132	142	153		
3	70	105	115	122	134	142	152			
4	82	117	127	134	146	154				
5	95	**130**	**140**	**147**	**159**					
6	102	137	**147**	154						
7	110	145	155							
8	110	145								

Now the table below shows the allocation and the outcomes for zone 3 and zone 4.'

Number of Salesmen	0	1	2	3	4	5	6	7		8
Zone 3	0	1	1	2	3	5	1	1	2	3
Zone 4	0	0	1	1	1	0	5	6	5	5
Return in Rs. × 1000	77	97	99	106	118	130	140	147	147	159

In third stage we combine both zones 1 & 2 outcomes and zones 3 and 4 outcomes.

Zones 1 and 2 and zones 3 and 4 combined.

Zones 1& 2 Salesmen →		(0,0) 0	(0,1) 1	(0,2) 2	(1,2) 3	(2,2) 4	(3,2) 5	(4,2) 6	(4,3) 7	(4,4)(3,5) 8
Return.		75	90	105	118	130	142	153	163	172
Zones 3 & 4 Salesmen	Return. ↓									
0(0,0)	77	**152**	**187**	182	195	207	219	230	240	247
1(1,0)	97	172	**187**	**202**	**215**	**302**	**239**	**243**	**260**	
2(1,1)	99	174	189	204	217	229	241	252		
3(2,1)	106	181	196	211	224	236	248			
4 (3,1)	118	193	208	223	236	248				
5(5,0)	130	205	220	235	248					
6(1,5)	140	215	230	245						
7(1,6) (2,5)	147	222	237							
8(3,5)	159	234								

Optimal allocation is:

Salesmen	0	1	2	3	4	5	6	7	8
Zone 1	0	0	1	0	1	2	3	4	4
Zone 2	0	0	0	2	2	2	2	2	3
Zone 3	0	1	1	1	1	1	1	1	1
Zone 4	0	0	0	0	0	0	0	0	0
Total return in Rs. × 1000	152	187	187	202	215	302	239	243	260

The above table shows that how salesmen are allocated to various zones and the optimal outcome for the allocation. **Maximum outcome is Rs. 260 × 1000.**

Note: Students may try different combinations, *i.e.* first combining zone 1 and zone 3 and then zones 2 and 4 and then combining both. Then also the optimal outcome will be same. OR add 1 and 2 zones, then add zone 3 and then zone 4 to it. Then also the optimal outcome will be same.

Problem 11.2.

The owner of a chain of four grocery stores has purchased six crates of fresh strawberries. The estimated probability distribution of potential sales of the strawberries before spoilage differs among the four stores. The following table gives the estimated total expected profit at each store, when it is allocated various numbers of crates:

Stores.

Number of Crates	1	2	3	4
0	0	0	0	0
1	4	2	6	2
2	6	4	8	3
3	7	6	8	4
4	7	8	8	4
5	7	9	8	4
6	1	10	8	4

For administrative reasons, the owner does not wish to split crates between stores. However he is willing to distribute zero crates to any of his stores.

Solution

Let the four stores be considered as four stages in dynamic programming formulation. The decision variables x_i ($i = 1, 2, 3$ and 4) denote the number of crates allocated to the i th stage. Let $f(x_i)$ be the expected profit from allocation of x_i crates to the store 'i', then the problem is:

Maximize $Z = f_1(x_1) + f_2(x_2) + f_3(x_3) + f_4(x_4)$ subject to

$x_1 + x_2 + x_3 + x_4 = 6$ and all $x_i \geq 0$

Store 3 →

Store 4 ↓

No. of crates		0	1	2	3	4	5	6
Profit. →		0	6	8	8	8	8	8
No. of crates	Profit. ↓							
0	0	0	**6**	**8**	8	8	8	8
1	2	2	3	**10**	10	10	10	
2	3	3	9	**11**	11	11		
3	4	4	10	**12**	**12**			
4	4	4	10	**12**				
5	4	4	10					
6	4	4						

Allocation for the first stage:

No. of crates	0	1	2	3	4	5	6	
Store 3	0	1	2	2	2	2	3	2
Store 4	0	0	0	1	2	3	3	4
Profit	0	6	8	10	11	12	12	

Store 2 ↓ Store 1 →

No. of crates		0	1	2	3	4	5	6
Profit →		0	4	6	7	7	7	7
No. of crates	Profit ↓							
0	0	**0**	**4**	**6**	7	7	7	7
1	2	2	**6**	**8**	9	9	9	
2	4	4	**8**	**10**	11	11		
3	6	6	**10**	**12**	12			
4	8	8	**12**	**14**				
5	9	9	13					
6	10	10						

No. of crates	0	1	2		3		4		5		6
Store 1	0	1	2	1	2	1	2	1	2	1	2
Store 2	0	0	0	1	1	2	2	3	3	4	4
Profit.	0	4	6	6	8	8	10	10	12	12	14

Stores 3 & 4

Stores 1 & 2 ↓

No. of crates		0 0,0	1 1,0	2 2,0	3 2,1	4 2,2	5 2,3	6 3 2
Profit →		0	6	8	10	11	12	12
No. of crates	↓ Profit							
0 (0,0)	0	0	6	8	10	11	12	12
1 (1,0)	4	4	10	12	14	15	16	
2 (2, 0), (1, 1)	6	6	12	14	16	17		
3 (2, 1), (1, 2)	8	8	14	16	18			
4 (2,2), (1, 3)	10	10	16	18				
5(2, 3), (1, 4)	12	12	18					
6 (2, 4)	14	14						

All the four stores combined at 3 rd stage.

No. of crates	0	1	2	3			4					5						6					
Store 1				2	1	1	2	1	2	1	1	2	1	2	1	2	1	2	1	2	1	2	1
Store 2				0	1	0	1	2	0	1	0	2	3	1	2	0	1	3	4	2	3	1	2
Store 3				1	1	2	1	1	2	2	2	1	1	2	2	2	2	1	1	2	2	2	2
Store 4				0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	1	1
Profit.	0	6	10	12			14					16						18					

Maximum profit is Rs. 18/-