INTRODUCTION AND PRELIMINARIES: SEQUENTIAL OPTIMIZATION

INTRODUCTION

In some complex problems, it will be advisable to approach the problem in a sequential manner in order to find the solution quickly. The solution is found out in multi stages. This is the basic approach behind dynamic programming. It works in a *"divide and conquer"* manner. The word "programming" in "dynamic programming" has no particular connection to computer programming at all. A program is, instead, the plan for action that is produced. In this lecture, the multistage decision process, its representation, various types and the concept of sub-optimization and principle of optimality are discussed.

In order to solve a multistage decision problem in sequence, we make use of recursive equations. These equations are fundamental to the dynamic programming. In this lecture, we will also learn how to formulate and solve recursive equations for a multistage decision process in a backward manner and also in a forward manner. In this lecture, we will be also discussing about problems with state variable taking continuous values and also problems with multiple state variables.

SEQUENTIAL OPTIMIZATION

In sequential optimization, a problem is approached by dividing it into smaller subproblems and optimization is done for these subproblems without losing the integrity of the original problem. Sequential decision problems are those in which decisions are made in multiple stages. These are also called multistage decision problems since decisions are made at a number of stages.

In multistage decision problems, an N variable problem is represented by N single variable problems. These problems are solved successively such that the optimal value of the original problem can be obtained from the optimal solutions of these N single variable problems. The N single variable problems are connected in series so that the output of one stage will be the input to the succeeding stage. This type of solution is called the serial multistage decision process.

For example, consider a water allocation problem to N users. The objective function is to maximize the total net benefit from all users. This problem can be solved by considering each

user separately and optimizing the individual net benefits, subject to constraints and then adding up the benefits from all users to get the total optimal benefit.

REPRESENTATION OF MULTISTAGE DECISION PROCESS

Consider a single stage decision process as shown in the figure 1.

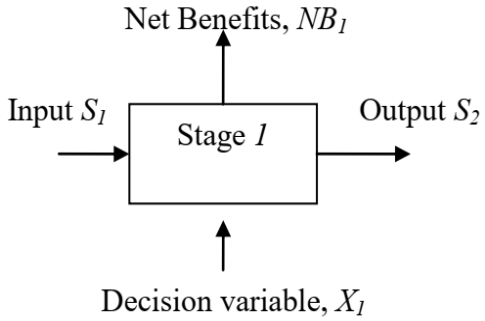

Fig 1. Single stage decision process

Let S_1 be the input state variable, S_2 be the output state variable, X_1 be the decision variable and NB_1 be the net benefits. The input and output are related by a transformation function expressed as,

$$S_2 = g(X_1, S_1)$$

Also since the net benefits are influenced by the decision variables and also the input variable, the benefit function can be expressed as

$$NB_1 = h(X_1, S_1)$$

Now, consider a serial multistage decision process consisting of T stages as shown in the figure 2.

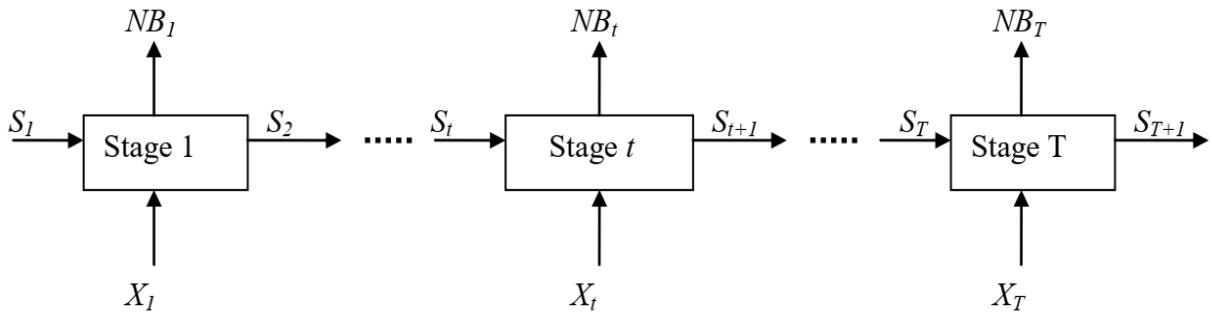

Fig 2. Multistage decision process

Here, for the t^{th} stage, the state transformation and the benefit functions are written as,

$$S_{t+1} = g(X_t, S_t)$$

$$NB_t = h(X_t, S_t)$$

The objective of this multistage problem is to find the optimum values of all decision variables $X_1, X_2, \ldots, X_T$ such that the individual net benefits of each stage that is expressed by some objective function, $f(NB_t)$ and the total net benefit which is expressed by $f(NB_1, NB_2, \ldots, NB_T)$ should be maximized. The application of dynamic programming to a multistage problem depends on the nature of this objective function i.e., the objective function should be separable and monotonic. An objective function is separable, if it can be decomposed and expressed as a sum or product of individual net benefits of each stage, i.e.,

either $f = \sum_{t=1}^{T} NB_t = \sum_{t=1}^{T} h(X_t, S_t)$

or $f = \prod_{t=1}^{T} NB_t = \prod_{t=1}^{T} h(X_t, S_t)$

An objective function is monotonic if for all values of a and b for which the value of the benefit function is $h(x_t = a, S_t) \geq h(x_t = b, S_t)$, then

$$f(x_1, x_2, \ldots, x_t = a, \ldots, x_T, S_{t+1}) \geq f(x_1, x_2, \ldots, x_t = b, \ldots, x_T, S_{t+1})$$

should be satisfied.

Types of multistage decision problems

A serial multistage problem such as shown, can be classified into three categories as *initial value problem, final value problem and boundary value problem.*

1. Initial value problem: In this type, the value of the initial state variable, S_1 is given.
2. Final value problem: In this, the value of the final state variable, S_T is given. A final value problem can be transformed into an initial value problem by reversing the procedure of computation of the state variable, S_t.
3. Boundary value problem: In this, the values of both the initial and final state variables, S_1 and S_T are given.

Concept of sub-optimization and principle of optimality

Bellman (1957) stated the principle of optimality which explains the process of suboptimality as:

"An optimal policy (or a set of decisions) has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."

Consider the objective function consisting of T decision variables $x_1, x_2, \ldots, x_T$,

$$f = \sum_{t=1}^{T} NB_t = \sum_{t=1}^{T} h(X_t, S_t)$$

and satisfying the equations,

$$S_{t+1} = g(X_t, S_t)$$

$$NB_t = h(X_t, S_t) \qquad \text{for } t = 1,2,\ldots,T$$

The concepts of suboptimization and principle of optimality are used to solve this problem through dynamic programming. To explain these concepts, consider the design of a water tank in which the cost of construction is to be minimized shown in figure 3. The capacity of the tank to be designed is given as K.

The main components of a water tank include (i) tank (ii) columns to support the tank and (iii) the foundation. While optimizing this problem to minimize the cost, it would be advisable to break this system into individual parts and optimizing each part separately instead of considering the system as a whole together. However, while breaking and doing suboptimization, a logical procedure should be used; otherwise this approach can lead to a poor solution. For example, consider the suboptimization of columns without considering the other two components. In order to reduce the construction cost of columns, one may use heavy concrete columns with less reinforcement, since the cost of steel is high. But while considering the suboptimization of foundation component, the cost becomes higher as the foundation should be strong enough to carry these heavy columns. Thus, the suboptimization of columns before considering the suboptimization of foundation will adversely affect the overall design.

In most of the serial systems as discussed above, since the suboptimization of last component does not influence the other components, it can be suboptimized independently. For the above problem, foundation can thus be suboptimized independently. Then the last two

components (columns and foundation) are considered as a single component and suboptimization is done without affecting other components. This process can be repeated for any number of end components. The process of suboptimization for the above problem is shown in the next page.

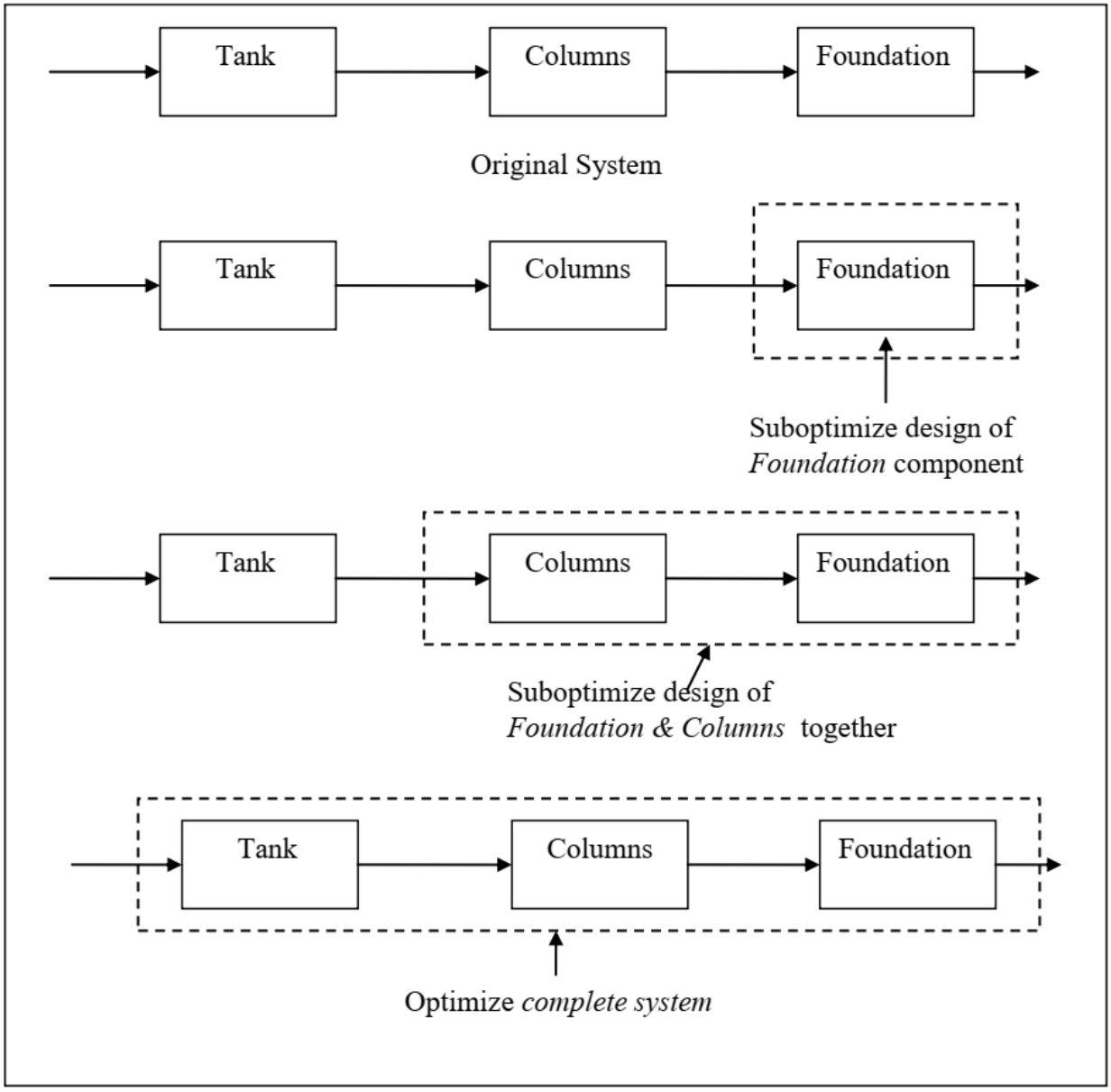

Fig 3. Design of a water tank

RECURSIVE EQUATIONS

Recursive equations are used to structure a multistage decision problem as a sequential process. Each recursive equation represents a stage at which a decision is required. In this, a series of equations are successively solved, each equation depending on the output values of the previous equations. Thus, through recursion, a multistage problem is solved by breaking it

into a number of single stage problems. A multistage problem can be approached in a backward manner or in a forward manner.

Backward recursion

In this, the problem is solved by writing equations first for the final stage and then proceeding backwards to the first stages. Consider the serial multistage problem.

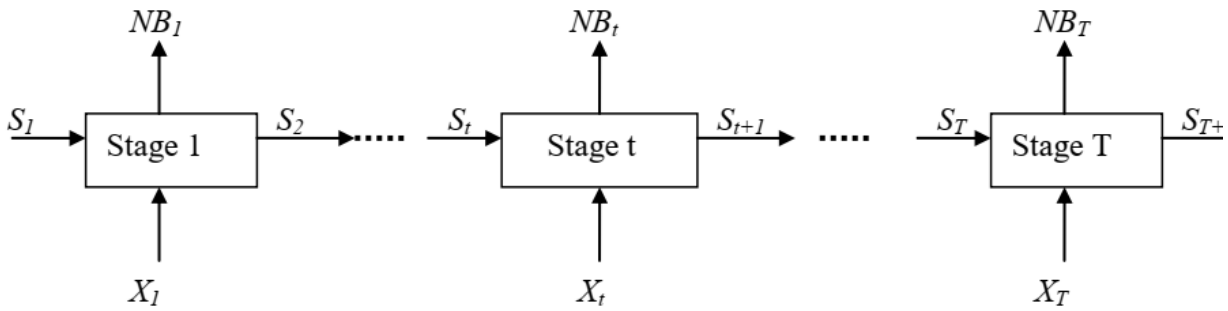

Fig. 4 Multistage problem

Suppose the objective function for this problem is

$$f = \sum_{t=1}^{T} NB_t = \sum_{t=1}^{T} h_t\left(X_t, S_t\right)$$
$$= h_1\left(X_1, S_1\right) + h_2\left(X_2, S_2\right) + \ldots + h_t\left(X_t, S_t\right) + \ldots + h_{T-1}\left(X_{T-1}, S_{T-1}\right) + h_T\left(X_T, S_T\right) \quad \ldots(1)$$

and the relation between the stage variables and decision variables are gives as

$$S_{t+1} = g(X_t, S_t), \quad t = 1,2,\ldots, T. \quad \ldots(2)$$

Consider the final stage as the first subproblem. The input variable to this stage is S_T. According to the principle of optimality, no matter what happens in other stages, the decision variable X_T should be selected such that $h_T\left(X_T, S_T\right)$ is optimum for the input S_T. Let the optimum value be denoted as f_T^*. Then,

$$f_T^*(S_T) = \underset{X_T}{opt}\ h_T\left(X_T, S_T\right) \quad \ldots(3)$$

Next, group the last two stages together as the second subproblem. Let f_{T-1}^* be the optimum objective value of this subproblem. Then, we have

$$f_{T-1}^*(S_{T-1}) = \underset{X_{T-1}, X_T}{opt}\ h_{T-1}\left(X_{T-1}, S_{T-1}\right) + h_T\left(X_T, S_T\right) \quad \ldots(4)$$

From the principle of optimality, the value of X_T should be to optimize h_T for a given S_T. For obtaining S_T, we need S_{T-1} and X_{T-1}. Thus, $f^*_{T-1}(S_{T-1})$ can be written as,

$$f^*_{T-1}(S_{T-1}) = \underset{X_{T-1}}{opt}\ h_{T-1}(X_{T-1}, S_{T-1}) + f^*_T(S_T) \quad \ldots(5)$$

By using the stage transformation equation, $f^*_{T-1}(S_{T-1})$ can be rewritten as,

$$f^*_{T-1}(S_{T-1}) = \underset{X_{T-1}}{opt}\ h_{T-1}(X_{T-1}, S_{T-1}) + f^*_T(g_{T-1}(X_{T-1}, S_{T-1})) \quad \ldots(6)$$

Thus, here the optimum is determined by choosing the decision variable X_{T-1} for a given input S_{T-1}. Eqn (4) which is a multivariate problem (second sub problem) is divided into two single variable problems as shown in eqns (3) and (6). In general, the $i+1^{th}$ subproblem (T-i^{th} stage) can be expressed as,

$$f^*_{T-i}(S_{T-i}) = \underset{X_{T-i},\ldots,X_{T-1},X_T}{opt} h_{T-i}(X_{T-i}, S_{T-i}) + \ldots + h_{T-1}(X_{T-1}, S_{T-1}) + h_T(X_T, S_T) \quad \ldots(7)$$

Converting this to a single variable problem,

$$f^*_{T-i}(S_{T-i}) = \underset{X_{T-i}}{opt}\ h_{T-i}(X_{T-i}, S_{T-i}) + f^*_{T-(i-1)}(g_{T-i}(X_{T-i}, S_{T-i})) \quad \ldots(8)$$

where $f^*_{T-(i-1)}$ denotes the optimal value of the objective function for the last i stages. Thus for backward recursion, the principle of optimality can be stated as, *no matter in what state of stage one may be, in order for a policy to be optimal, one must proceed from that state and stage in an optimal manner.*

Forward recursion

In this approach, the problem is solved by starting from the stage 1 and proceeding towards the last stage. Consider the same serial multistage problem with the objective function as given below

$$f = \sum_{t=1}^{T} NB_t = \sum_{t=1}^{T} h_t(X_t, S_t)$$
$$= h_1(X_1, S_1) + h_2(X_2, S_2) + \ldots + h_t(X_t, S_t) + \ldots + h_{T-1}(X_{T-1}, S_{T-1}) + h_T(X_T, S_T) \quad \ldots(9)$$

and the relation between the stage variables and decision variables are gives as

$$S_t = g'(X_{t+1}, S_{t+1}) \qquad t = 1,2,\ldots,T \qquad \ldots(10)$$

where S_t is the input available to the stages *1* to *t*.

Consider stage 1 as the first subproblem. The input variable to this stage is S_1. The decision variable X_1 should be selected such that $h_1(X_1, S_1)$ is optimum for the input S_1.

The optimum value f_1^* can be written as

$$f_1^*(S_1) = \underset{X_1}{opt}\, h_1(X_1, S_1) \quad \ldots(11)$$

Now, group the first and second stages together as the second subproblem. The objective function f_2^* for this subproblem can be expressed as,

$$f_2^*(S_2) = \underset{X_2, X_1}{opt}\, h_2(X_2, S_2) + h_1(X_1, S_1) \quad \ldots(12)$$

So for calculating the value of S_2, we need S_1 and X_1. Thus,

$$f_2^*(S_2) = \underset{X_2}{opt}\, h_2(X_2, S_2) + f_1^*(S_1) \quad \ldots(13)$$

By using the stage transformation equation, $f_2^*(S_2)$ can be rewritten as,

$$f_2^*(S_2) = \underset{X_2}{opt}\, h_2(X_2, S_2) + f_1^*(g_2'(X_2, S_2)) \quad \ldots(14)$$

Thus, here through the principle of optimality the dimensionality of the problem is reduced from two to one. In general, the i^{th} subproblem can be expressed as,

$$f_i^*(S_i) = \underset{X_1, X_2, \ldots, X_i}{opt}\, h_i(X_i, S_i) + \ldots + h_2(X_2, S_2) + h_1(X_1, S_1) \quad \ldots(15)$$

Converting this to a single variable problem,

$$f_i^*(S_i) = \underset{X_i}{opt}\, h_i(X_i, S_i) + f_{(i-1)}^*(g_i'(X_i, S_i)) \quad \ldots(16)$$

where f_i^* denotes the optimal value of the objective function for the first i stages. The principle of optimality for forward recursion is that *no matter in what state of stage one may be, in order for a policy to be optimal, one had to get to that state and stage in an optimal manner.*

Computational procedure

Consider the serial multistage problem and the recursive equations developed for backward recursion.

The objective function for this problem is

$$f = \sum_{t=1}^{T} NB_t = \sum_{t=1}^{T} h_t\left(X_t, S_t\right) \\ = h_1\left(X_1, S_1\right) + h_2\left(X_2, S_2\right) + \ldots + h_t\left(X_t, S_t\right) + \ldots + h_{T-1}\left(X_{T-1}, S_{T-1}\right) + h_T\left(X_T, S_T\right) \quad (17)$$

Considering first subproblem i.e., the last stage, the objective function is

$$f_T^*(S_T) = \underset{X_T}{opt}\, h_T\left(X_T, S_T\right) \quad (18)$$

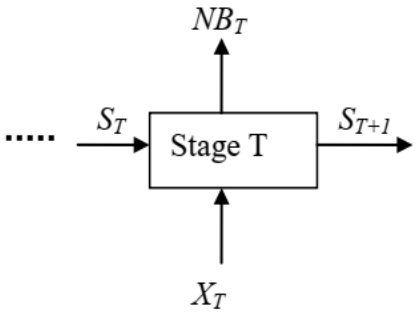

Fig. 5 Last stage

The input variable to this stage is S_T. The decision variable X_T and the optimal value of the objective function f_T^* depend on the input S_T. At this stage, the value of S_T is not known. S_T can take a range of values depending upon the value taken by the upstream components. To get a clear picture of the suboptimization at this stage, S_T is solved for all possible range of values and the results are entered in a graph or table. This table also contains the calculated optimal values of X_T^*, S_{T+1} and also f_T^*. A typical table showing the results from the suboptimization of stage 1 is shown below.

Table 1

Sl no	S_T	X_T^*	f_T^*	S_{T+1}
1	-	-	-	-
-	-	-	-	-
-	-	-	-	-

Now, consider the second subproblem by grouping the last two components as shown in figure 6.

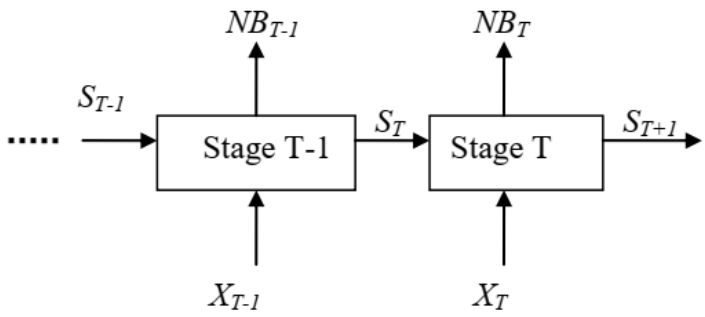

Fig. 6 Grouping of last two components

The objective function can be written as

$$f_{T-1}^*(S_{T-1}) = \underset{X_{T-1},X_T}{opt}\ h_{T-1}(X_{T-1},S_{T-1}) + h_T(X_T,S_T) \tag{19}$$

$f_{T-1}^*(S_{T-1})$ can also be written as,

$$f_{T-1}^*(S_{T-1}) = \underset{X_{T-1}}{opt}\ h_{T-1}(X_{T-1},S_{T-1}) + f_T^*(S_T) \tag{20}$$

Here also, a range of values are considered for S_{T-1}. All the information of first subproblem can be obtained from Table 1. Thus, the optimal values of X_{T-1}^* and f_{T-1}^* are found for these range of values. The results thus calculated can be shown in Table 2.

Table 2

Sl no	S_{T-1}	X_{T-1}^*	S_T	$f_T^*(S_T)$	f_{T-1}^*
1	-	-	-	-	
-	-	-	-	-	
-	-	-	-	-	

In general, if suboptimization of $i+1^{th}$ subproblem (T-i^{th} stage) is to be done, then the objective function can be written as

$$\begin{aligned} f_{T-i}^*(S_{T-i}) &= \underset{X_{T-i},\ldots,X_{T-1},X_T}{opt} h_{T-i}(X_{T-i},S_{T-i}) + \ldots + h_{T-1}(X_{T-1},S_{T-1}) + h_T(X_T,S_T) \\ &= \underset{X_{T-i}}{opt}\ h_{T-i}(X_{T-i},S_{T-i}) + f_{T-(i-1)}^* \end{aligned} \tag{21}$$

At this stage, the suboptimizaiton has been carried out for all last i components. The information regarding the optimal values of i^{th} subproblem will be available in the form of a table. Substituting this information in the objective function and considering a range of S_{T-i}

values, the optimal values of f^*_{T-i} and X^*_{T-i} can be calculated. The table showing the suboptimization of $i+1^{th}$ subproblem can be as shown below.

Table 3

Sl no	S_{T-i}	X^*_{T-i}	$S_{T-(i-1)}$	$f^*_{T-(i-1)}(S_{T-(i-1)})$	f^*_{T-i}
1	-	-	-	-	
-	-	-	-	-	
-	-	-	-	-	

This procedure is repeated until stage 1 is reached, i.e., T^{th} subproblem. (figure 7)

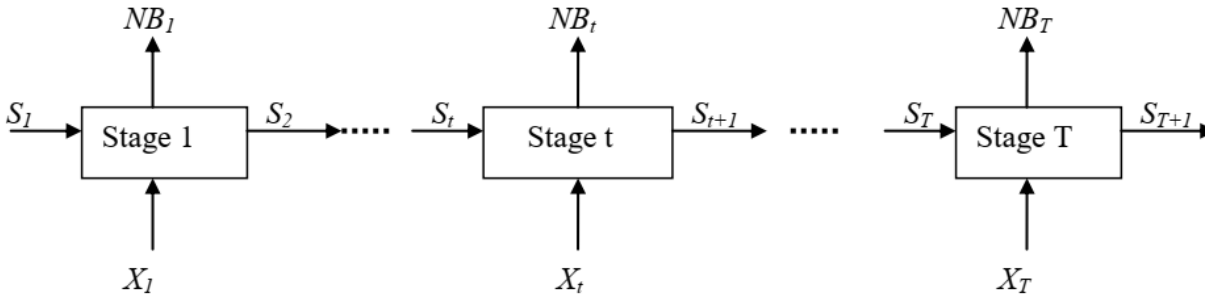

Fig. 7

Here, for initial value problems, only one value S_1 need to be analyzed.

After completing the suboptimization of all the stages, we need to retrace the steps through the tables generated to find the optimal values of X. In order to do this, the T^{th} subproblem gives the values of X_1^* and f_1^* for a given value of S_1 (since the value of S_1 is known for an initial value problem). Use the transformation equation $S_2 = g(X_1, S_1)$, to calculate the value of S_2^*, which is the input to the 2nd stage (T-1^{th} subproblem). Then, from the tabulated results for the 2nd stage, the values of X_2^* and f_2^* are found out for the calculated value of S_2^*. Again use the transformation equation to find out S_3^* and the process is repeated until the 1st subproblem or T^{th} stage is reached. Finally, the optimum solution vector is given by $X_1^*, X_2^*, ..., X_T^*$.

Discrete versus Continuous Dynamic Programming

In a dynamic programming problem, when the number of stages tend to infinity then it is called a continuous dynamic programming problem. It is also called an infinite-stage

problem. Continuous dynamic programming model is used to solve continuous decision problems. The classical method of solving continuous decision problems is by the calculus of variations. However, the analytical solutions, using calculus of variations, cannot be generally obtained, except for very simple problems. The infinite-stage dynamic programming approach, on the other hand provides a very efficient numerical approximation procedure for solving continuous decision problems.

The objective function of a conventional discrete dynamic programming model is the sum of individual stage outputs. If the number of stages tends to infinity, then summation of the outputs from individual stages can be replaced by integrals. Such models are useful when infinite number of decisions have to be made in finite time interval.

MULTIPLE STATE VARIABLES

In the problems previously discussed, there was only one state variable S_t. However, there will be problems in which one need to handle more than one state variable. For example, consider a water allocation problem to n irrigated crops. Let S_i be the units of water available to the remaining n-i crops. If we are concerned only about the allocation of water, then this problem can be solved as a single state problem, with S_i as the state variable. Now, assume that L units of land are available for all these n crops. We want to allocate the land also to each crop after considering the units of water required for each unit of irrigated land containing each crop. Let R_i be the amount of land available for n-i crops. Here, an additional state variable R_i is to be included while suboptimizing different stages. Thus, in this problem two allocations need to be made: water and land.

The figure below shows a single stage problem consisting of two state variables, S_1 & R_1.

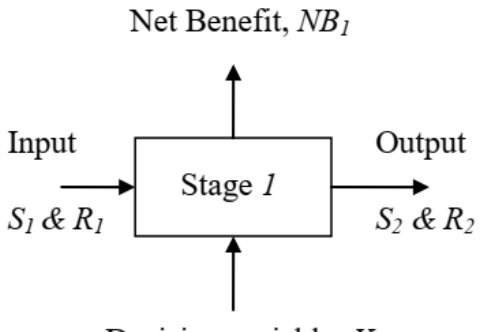

Fig. 8 Single stage problem with two variables

In general, for a multistage decision problem of T stages containing two state variables S_t and R_t, the objective function can be written as

$$f = \sum_{t=1}^{T} NB_t = \sum_{t=1}^{T} h\left(X_t, S_t, R_t\right)$$

where the transformation equations are given as

$$S_{t+1} = g(X_t, S_t) \qquad \text{for } t = 1,2,\ldots, T$$

$$\& \qquad R_{t+1} = g'(X_t, R_t) \qquad \text{for } t = 1,2,\ldots, T$$

CURSE OF DIMENSIONALITY

Dynamic programming has a serious limitation due to dimensionality restriction. As the number of variables and stages increase, the number of calculations needed increases rapidly thereby increasing the computational effort. If the number of stage variables is increased, then more combinations of discrete states should be examined at each stage. For a problem consisting of 100 state variables and each variable having 100 discrete values, the suboptimization table will contain 100^{100} entries. The computation of this one table may take 100^{96} seconds (about 100^{92} years) even on a high speed computer. Like this 100 tables have to be prepared, which explains the difficulty in analyzing such a big problem using dynamic programming. This phenomenon is known as *"curse of dimensionality" or "Problem of dimensionality"* of multiple state variable dynamic programming problems

BIBLIOGRAPHY / FURTHER READING:

1. Bellman, R., *Dynamic Programming*, Princeton University Press, Princeton, N.J, 1957.
2. Hillier F.S. and G.J. Lieberman, *Operations Research*, CBS Publishers & Distributors, New Delhi, 1987.
3. Loucks, D.P., J.R. Stedinger, and D.A. Haith, *Water Resources Systems Planning and Analysis*, Prentice-Hall, N.J., 1981.
4. Rao S.S., *Engineering Optimization – Theory and Practice*, Fourth Edition, John Wiley and Sons, 2009.
5. Taha H.A., *Operations Research – An Introduction*, 8th edition, Pearson Education India, 2008.
6. Vedula S., and P.P. Mujumdar, *Water Resources Systems: Modelling Techniques and Analysis*, Tata McGraw Hill, New Delhi, 2005.