DYNAMIC PROGRAMMING

by

Dr. S K Jain,
Scientist E, NIH, Roorkee.

INTRODUCTION

Dynamic programming is an enumerative technique developed by Bellman in 1953. This technique was developed to optimize a problem which can be represented as a multistage decision process. The entire formulation of dynamic programming is based upon the Bellman's principle of optimality, Fig. 1. According to this principle an optimal policy has the property that whatever the initial state and decisions are, the remaining decisions must constitute an optimal policy with respect to the state resulting from the first decision. The proof can be obtained by contradiction. If the optimal path for going from A to C is I-II then the optimal path from B to C will be II and not II'.

To decompose a general problem into stages with decisions required at each stage, the value of every stage should satisfy the separability condition and the monotonicity condition (Nemhuser,1966). The validity of DP regarding separability can be extended by increasing the number of states, but this extension is achieved at the cost of increased computations. For example, reservoir inflows are Markovian rather than random in the transition equation. If the returns are additive, multiplicative, or of the minimax or maximin variety, it is sufficient to check that the returns are independent.

Where there is no special reason for choosing either backward or forward formulation, the backward recurrence is normally used. The procedure of making first a backward and then a forward pass is convenient especially in problem involving time, as it gives the optimal policy in the chronological order; in stochastic problems backward recurrence is essential, since each stage depends on the results of the former stage. But, forward recurrence is advantageous when a deterministic problem has to be solved several times with different planning horizons. This may occur because a plan is periodically reviewed or where the appropriate horizon is unknown and a sensitivity analysis is undertaken. The value can be extended forward in time without repeating previous calculations.

Constraints which restrict only the state or decision space are advantageous in DP because they reduce the amount of computation. By contrast, state and decision space constraints can cause considerable procedural difficulties for other optimization techniques. However, when DP is applied to a multiple reservoir system, the usefulness of the technique is limited by the so-called curse of dimensionality which is a strong function of the number of the state variables. For computational efficiency, problems should have no more than a few state variables at a

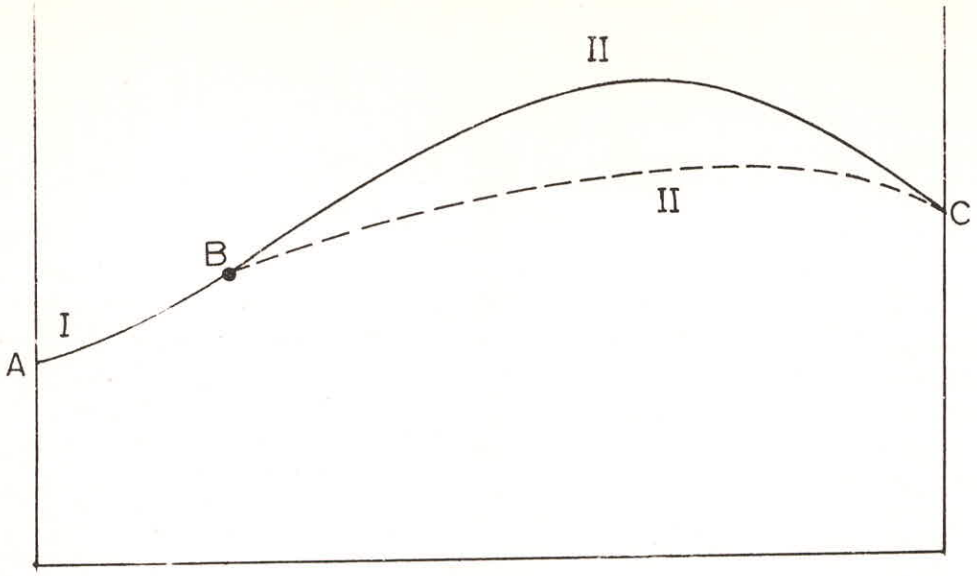

FIG. 1 ILLUSTRATION OF THE PRINCIPLE OF OPTIMALITY

time.All method of dimensionality reduction involve decomposition into subsystems and the use of iterative procedures. Also, there are difficulties in developing stochastic DP for a multiple reservoir system where serial and cross correlations are prevalent between streamflows. These difficulties could be partially overcome with the combined use of deterministic DP and simulation.

In a Dynamic Programming problem formulation, the dynamic behaviour of the system is expressed by using three types of variables, as described below :

a. State variables - which define the condition of the system. For example, in studies dealing with reservoirs, the amount of water stored in the reservoir may represent its state.

b. Stage variables - which define the order in which events occur in the system. Most commonly, time is taken to be the stage variable. There must be a finite number of possible states at each stage.

c. Control variables - which represent the controls applied at a particular stage and transform the state of the system. For the reservoir operation problem, the release of water from the reservoir is a typical decision variable.

With each state transformation, a return is associated which may either represent benefits or costs. The state of reservoir will be transformed by releasing a certain amount of water from it. This water can be used for some useful purpose like irrigation and will lead to monetary returns. The water released from a reservoir may also cause flood damages downstream and hence a cost can be associated with these damages. The problem is to find the control variables which optimize the returns. Typically the benefits are maximized and the costs are minimized. The optimal decision made at a particular stage is independent of decisions made at previous stage given the current state of the system. A set of decisions for each time period is called a policy. The particular policy which optimizes the objective function is called the optimal policy. The set of states which result from the application of a policy is called the state trajectory.

The dynamic behaviour of the system is expressed by an equation known as the system equation. It can be written in discrete form as :

$$s(t+1) = f[(s(t), u(t), t] \qquad t = 1,2.....N$$

where s(t) is the state variable at time t, u(t) is the control applied at time instant t, which last for a duration t and f is the given functional form.

The state of the system at any stage t should lie in the domain of admissible states at that stage. Similarly the control at any stage should also lie in the admissible domain at that

stage :

$$s(t) \in S(t), \qquad u(t) \in U(t)$$

where S(t) and U(t) are the domains of admissible states and controls at stage t.

THE SOLUTION ALGORITHM

Let R[s(t), u(t), t] be the return obtained if the system is at state s(t) at stage t and the control u(t) is applied at instant t lasting for a duration t. Further, let $F^*[s(N),N]$ be the sum of returns from application of controls from some initial stage at t = 0 to final stage at t = N. The objective of maximizing the sum of returns from the system can be expressed as

$$\text{Max } F[s(N), N]$$

Let the state of system at t = 0, $s(0) \in S(0)$ is known and the returns F[s(0),0] are also known. Let $F^*[s(0),0]$ be the optimum value of these returns. Now consider the first stage (of duration t). The optimal return for this period is given by

$$F^*[s(1),1] = \max_{u(0) \in U(0)} R[s(0),u(0),0] + F^*[s(0),0]$$

This equation is solved for each discrete level of state at t=1 as a function of control variables u(0). To do this, the state is discretized into a number of discrete levels (refer Fig. 2). Now a particular lattice point is chosen and all the admissible levels of decision variables which lead to this state are chosen. For each of these decision variables, the return F[s(1),1] is calculated. The maximum among these returns given the value of $F^*[s(1),1]$. This computation is repeated for each discrete value of s(1) and the results are stored.

The computations are performed in similar fashion for stage 2,3....N. The recursive equation for any stage t can be written as

$$F[s(t),t] = \max_{u(t-1) \in U(t-1)} R[s(t-1),u(t-1),t-1] + F[s(t-1),t-1]$$

Thus at the end of N stage, the values of $F^*[s(t),t]$, t=1,2...N are available. The optimal value of control variables or the optimal policy is obtained by tracing back the values of returns, corresponding to those states which satisfy the initial and final values and the constraints. The optimal state trajectory can be determined by using the system equation once the optimal policy is known.

The above computational scheme of dynamic programming is known as the forward algorithm since the computations start at the initial value of the state variable at stage 1 and move forward. In contrast to this, the computations can also commence at the

final value of state variable at the last stage and can move backwards. The optimal policy is retrieved by tracing forward from the returns. This algorithm is called the backward algorithm.

Let us consider the problem of determination of optimum release from a reservoir. If DP is applied for the determination of reservoir release, the state variable is the storage and the decision variable is the release. The stage is represented by the time period i. The stage-to-stage transformation is characterized by the characterized by the continuity equation,

$$S_{i+1} = S_i + I_i - R_i - e_i$$

subject to:

$$S_{min} \leq S_{i+1} \leq S_{max}$$

Suppose an objective function J(S, R) has been chosen for maximization. Note that J in general is a function of release as well as storage. A typical forward DP recursive equation can be written as

$$f_{i+1}(S_{i+1}) = \max_{R_i} [\, J\,(R_i, \bar{S}_i) + f_i(S_i)\,]$$

S_o = given initial storage, and

i = 0, 1, 2, ..., T.

The state variable (storage) is generally discretized into a number of feasible states shown in Fig. 2.

Suppose that the inflow sequence is given and the evaporation term e_i is temporarily ignored, the continuity equation now becomes

$$S_{i+1} = S_i + I_i - R_i$$

If S_{i+1} and S_i are chosen, R_i can be directly computed from the above continuity equation. The optimization is over the proper choices of R_i's. The problem of interpolation is avoided, since the R_i's are computed by fixing the states S_{i+1} and S_i. Solutions are imbedded in the discretized states. The infeasible transitions are discarded in the solution process.

The inclusion of the evaporation term e_i poses no difficulty, since evaporation is a function of the average storage $\bar{S}_i$ which is equal to $(S_{i+1} + S_i)/2$.

The recursive equation is carried out until the final stage T

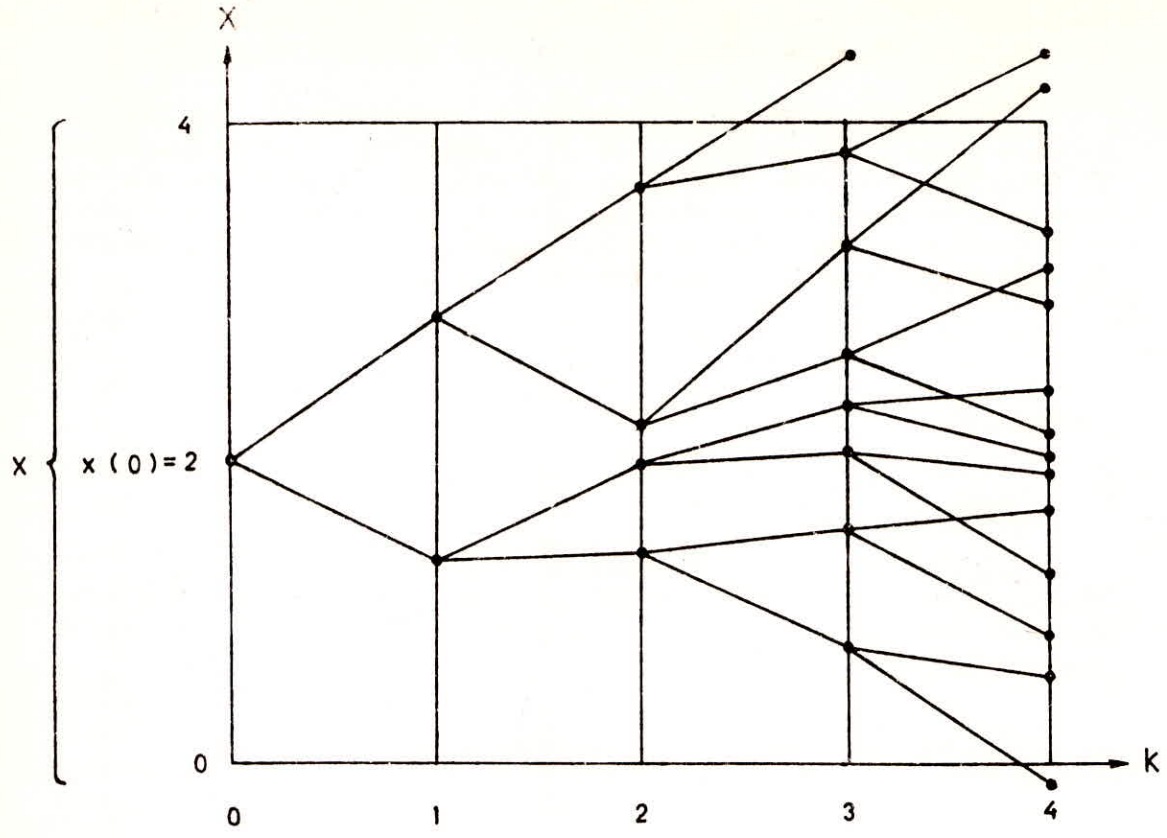

FIG. 2 TREE GENERATED BY ENUMERATION

is reached. The optimal solutions can then be traced back to determine the consequent release and storages.

ADVANTAGES AND DISADVANTAGES OF DYNAMIC PROGRAMMING

Dynamic programming is essentially an enumerative technique which is specially suited to multistage decision problems. There are a number of advantages in using this technique for analysis of a water resources system. Some of the advantages are :

i) The dynamic programming formulation is same for linear as well as nonlinear problems. Thus, no extra effort is required for nonlinear problems. This property is very useful in case of water resources systems since many related problems can not be realistically linearized.

ii) The incorporation of constraints in linear and nonlinear programming problems is more difficult than in dynamic programming problems. In case of dynamic programming, the constraints serve a useful purpose. They do limit the feasible region and thus many lead to reduction in computational time requirement.

iii) The stochastic nature of a problem can be easily considered in the dynamic programming formulation. The algorithm developed for a deterministic problem does not have to be significantly changed to incorporate stochasticity. This is in contrast with other techniques where incorporation of stochasticity requires too much change in the algorithm and significant increase in computational time.

Besides the above advantages, there are some disadvantages in using the dynamic programming formulation. These include:

i) The dynamic programming is not basically tailored in such a fashion that generalized programs can be written using it. Thus a new computer program has to be developed or an existing program has to be significantly modified and tested for each new application of the technique. On the other hand standard computer programs are widely available for the linear programming.

ii) It was stated above that to solve a particular problem, the state and control variables are discretized at each stage and these discretized values are then used. This approach is known as the Discrete Dynamic Programming(DDP) technique in which the state and control spaces are discretized by finite sets of vectors. For each stage and state, the continuous variables are replaced by the discrete node points and these values have to be stored in the computer memory from where they can be drawn as and when required. The number of discretized values goes on increasing with the fineness of the discretization. For large problems, the memory requirement becomes a major limitation. This requires judicious choice to be made for the accuracy requirement, computer memory available and computational time available.

Several techniques have been proposed by different investigators to reduce the dimensionality problem associated with

analysis of water resources systems using dynamic programming technique. These include the Incremental DP (IDP) and the Discrete Differential Dynamic Programming (DDDP). The basic approach in these techniques is the same and their are only minor differences regarding the increments in state and stage variables. It may be noted that these schemes are some sort of 'successive approximation' schemes. An initial estimate of the policy is made and this is used to construct an improved estimate. This improved estimate becomes the input to the next stage and so on until some convergence criterion is satisfied. The scheme can not assure the global optimum and may converge to a local optima. However, by starting from different initial solutions, the possibility of finding the global optimum is increased.

DISCRETE DIFFERENTIAL DYNAMIC PROGRAMMING

The Discrete Differential Dynamic Programming proposed by Heidari et al (1971) is an iterative procedure in which the recursive equation of dynamic programming is solved within a restricted set of quantized values of the state variables. The optimal solution is obtained by gradually improving the initial solution. This technique is particularly suitable for invertible systems. A system is called invertible if for that system, the order of the state vector is equal to the order of the control vector. Thus the knowledge of stage variables enables one to compute the decision variables. The water resources systems are mostly invertible. For example, assuming that the inflows to a reservoir are known, the releases from it can be determined if the states of the reservoir at different times are known.

The DDDP computations start with either a known stage trajectory or a known policy. Because of the property of invertibility the knowledge of one variable enables the determination of the other. The initial values must always satisfy the constraints.

Now a set of incremental values of stage variable is assumed. When these incremental values are added and subtracted from the trial trajectory at a particular stage, a subdomain is formed around the trial trajectory. This subdomain is called a corridor. At this stage optimization is performed constrained to this corridor and a better value of the trajectory is found. For the next iteration, this trajectory is considered as the trial trajectory. The computations are performed by varying the composition of the corridor in such a way that the algorithm converges towards the optimal solution.

One such corridor is shown in Fig. 3. In this case the trial trajectory lies at the centre of the corridor though this is not a necessary condition. More than one quantized states on either side of the trajectory may be chosen but the choice of three quantized states at each stage is most suitable for computational efficiency.

To obtain good convergence, two criterion were suggested by

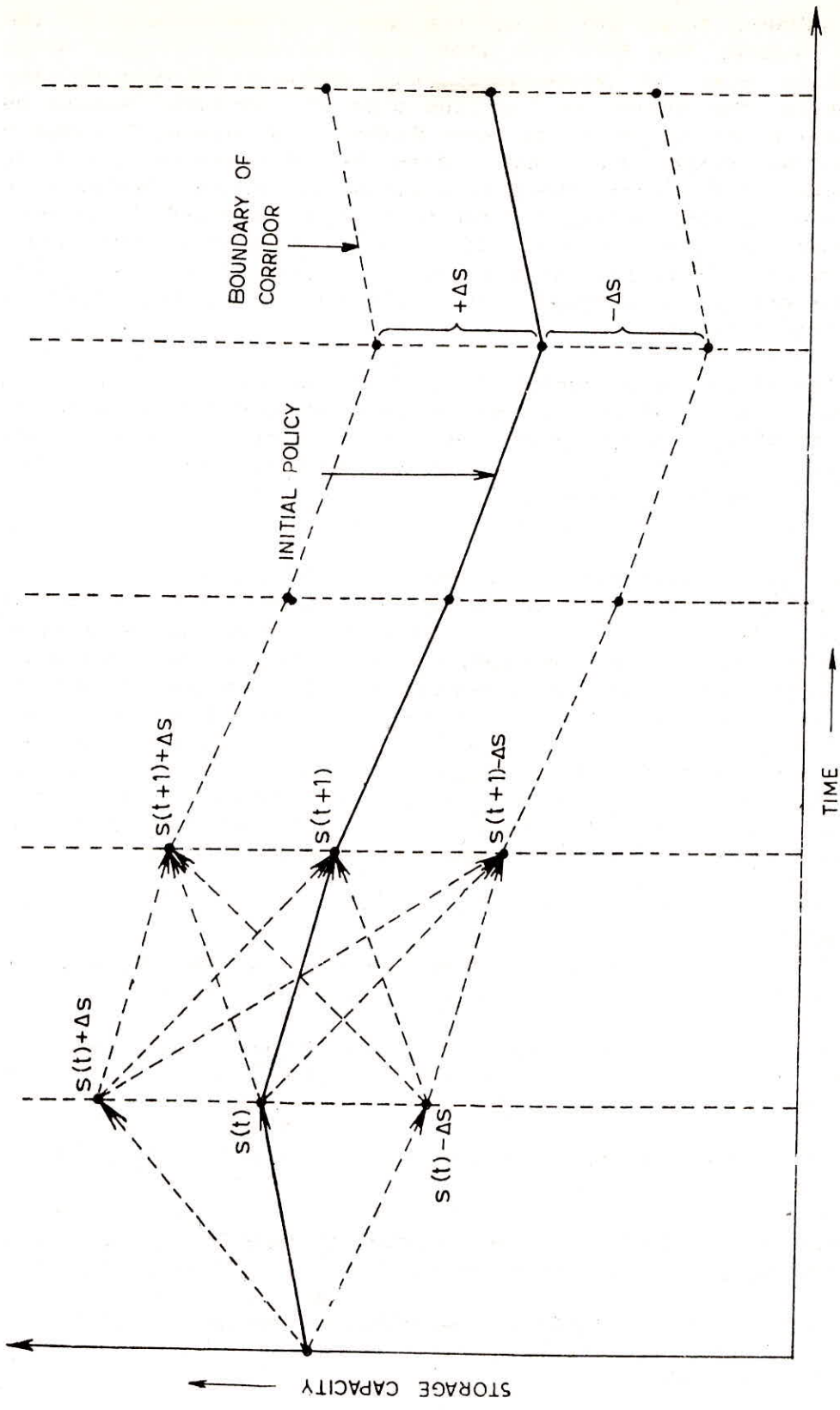

FIGURE- 3 DISCRETE DIFFERENTIAL DYNAMIC PROGRAMMING

Hall (1968). These are guidelines about the increments to the state vector. The first is that the increments to the state variables must be kept small and constant throughout any iteration. The second is that the size of increments should be reduced as the iterations proceed. However, the size of increments should be chosen such that entire feasible region could be inspected if required. There is a strong correlation between the number of iterations required for good convergence and the size of increments at each iteration. It was also suggested by Yeh (1982) that several iterations with a small increment should be allowed at the end of each computation cycle to improve the value of objective function

The technique of dynamic programming has been described in a number of excellent texts, some of these are given in references. Yakowitz(1982) has reviewed the applications of DP to water resources. Yeh(1985) provides an excellent review of applications of DP to reservoir operation.

Incremental DP with Successive Approximations (IDPSA)

Another way of alleviating the problem of "curse of dimensionality" is by using Bellman's concept of successive approximations which decomposes an original multiple-state variable DP into a series of subproblems of one state variable in such a manner that the sequence of optimizations over the subproblems converges to the solution of the original problem. This technique has also been applied to solve problems involving multiple reservoirs. It has also been extended to a more generalized case of higher-level combinations, such as two-at-a-time or three-at-a- time combinations.

REFERENCES

Hall,W.A., and J.A. Dracup, Water Resources Systems Engineering, Tata McGraw-Hill Publishing Company, New Delhi, 1979.

Loucks, D.P., J.R. Stedinger, and D.A. Haith, Water Resources Systems Planning and Analysis, Prentice Hall Inc., New Jersey, 1981.

Rao, S.S., Optimization, Theory and Practice, Wiley Eastern, 1979.

Yakowitz,S., "Dynamic programming applications in water resources", Water Resources Research, 18(4), 673-696, 1982.

Yeh, William W-G., "Reservoir management & operation models : A state of the art review", Water Resources Research, 21(12), 1797-1818, 1985.

***